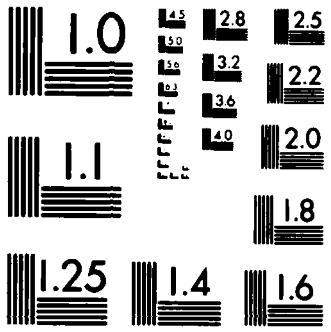
AD-A150 055 COST ESTIMATION TECHNIQUES FOR C3I SYSTEM SOFTWARE(U) 1/2
EDDINS-EARLES CONCORD MA M EDDINS-EARLES JUL 84
RADC-TR-84-156 F30602-83-C-0184

UNCLASSIFIED F/G 15/4 NL


1.0
4.5
5.0
5.6
6.3
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.1
1.8
1.25
1.4
1.6

(12)

RADC-TR-84-156
Final Technical Report
uly 1984

AD-A150 055

COST ESTIMATION TECHNIQUES FOR C^3I SYSTEM SOFTWARE

Eddins-Earles

Mary Eddins-Earles

This effort was funded under DoD Small Business Innovative Research Program (SBIR)

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

RADC-TR-84-156 has been reviewed and is approved for publication.

APPROVED: Andrew J. Chruscicki

ANDREW J. CHRUSCICKI
Project Engineer

APPROVED: Raymond P. Urtz Jr

RAYMOND P. URTZ, JR.
Technical Director
Command and Control Division

FOR THE COMMANDER: [illegible]

DONALD A. BRANTINGHAM
Acting Chief, Plans Office

AD-A150055

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS: N/A
2a. SECURITY CLASSIFICATION AUTHORITY: N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): N/A	5. MONITORING ORGANIZATION REPORT NUMBER(S): RADC-TR-84-156
6a. NAME OF PERFORMING ORGANIZATION: Eddins-Earles; 6b. OFFICE SYMBOL (If applicable): N/A	7a. NAME OF MONITORING ORGANIZATION: Rome Air Development Center (COEE)
6c. ADDRESS (City, State and ZIP Code): 89 Lee Drive, Concord MA 01742	7b. ADDRESS (City, State and ZIP Code): Griffiss AFB NY 13441
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Rome Air Development Center; 8b. OFFICE SYMBOL (If applicable): COEE	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: F30602-83-C-0184
8c. ADDRESS (City, State and ZIP Code): Griffiss AFB NY 13441	10. SOURCE OF FUNDING NOS.: PROGRAM ELEMENT NO. 65502F; PROJECT NO. 3005; TASK NO. RA; WORK UNIT NO. 01

11. TITLE (Include Security Classification): COST ESTIMATION TECHNIQUES FOR C^{3}I SYSTEM SOFTWARE

12. PERSONAL AUTHOR(S): Mary Eddins-Earles

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
Final	FROM Oct 83 TO May 84	July 1984	106

16. SUPPLEMENTARY NOTATION: This effort was funded under the DoD Small Business Innovative Research Program (SBIR)

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	
09	02	18.6	Software Cost Estimation
			Parametric Models
			Functional Sizing Data Base

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

This research developed the concept and computer programming requirements for a software life cycle cost estimating system which included methodology for the establishment of a data-base for Command, Control, Communications and Intelligence (C^{3}I) system sizing.

The proposed system uses the cost estimating relationships of the COCOMO model and generic files of baseline C^{3}I software designs for aid in sizing the number of source instructions required for a new design. The computer programming requirements are developed for a user-friendly interactive program. They permit computer program configuration items (CPCIs) to be designed by choosing computer program components (CPCs) from a stored library of functionally structured computer program modules. They permit CPCs to be designed by choosing generic modules of code from a similar stored library. They permit iterative life cycle cost estimates to be made at each level of the software breakdown structure with automatic re-computation with input changes.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED [✓] SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL: Andrew J. Chruscicki	22b. TELEPHONE NUMBER (Include Area Code): 315-330-4654	22c. OFFICE SYMBOL: RADC (COEE)

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

TABLE OF CONTENTS

SECTION PAGE

1. INTRODUCTION.. 1
2. MODEL RESEARCH.. 5
2.1 COCOMO... 5
2.1.1 Basic Regression................................... 5
2.1.1.1 Adaptation of Software........................... 7
2.1.1.2 Software Maintenance............................. 8
2.1.1.3 Computer Time.................................... 9
2.1.1.4 Clerical Effort.................................. 10
2.1.2 Intermediate Model................................. 10
2.1.3 Detailed Model..................................... 12
2.2 WICOMO.. 14
3. SIZING RESEARCH....................................... 16
3.1 Analogous Sizing..................................... 16
3.1.1 Aerospace Method................................... 16
3.1.2 Grumman Method..................................... 17
3.1.3 HARDMAN Model...................................... 20
3.2 Core Requirements Sizing............................. 23
3.3 PERT Sizing.. 24
3.4 Software Science..................................... 25
4. DATA RESEARCH... 26
4.1 DACS Data.. 27
4.1.1 DACS Productivity Dataset.......................... 27
4.1.2 Reliability Dataset................................ 29
4.1.3 NASA/SEL Dataset................................... 29
4.1.4 Verification and Validation Dataset................ 31
4.1.5 Operations and Maintenance Dataset................. 32
4.2 Aerospace Data....................................... 33
4.3 ESD Project Data..................................... 35
4.3.1 Projects... 35
4.3.2 SARE Data Collection Methodology................... 37
5. CONCEPT DEVELOPMENT................................... 48
5.1 Selected Features.................................... 48
5.2 Dataset Formats...................................... 49
5.3 Dataset Inputs....................................... 55
5.4 Cost Estimating Equations............................ 57
5.5 Help Screens... 60
6. INTERACTIVE COMPUTER PROGRAM.......................... 73
6.1 Structure.. 73
6.2 Logic.. 75
6.3 Functions.. 76
6.4 Application.. 79
7. SIZING LIBRARIES...................................... 81
REFERENCES... 93
BIBLIOGRAPY.. 95

LIST OF FIGURES

FIGURE		PAGE
2-1	WICOMO Interactive Screen Format	15
3-1	"SOFCOST" Search Display Example	19
3-2	Equipment Library File	22
4-1	Software Work Breakdown Structure for Aerospace Data	34
4-2	CPCI Work Breakdown Structure Elements	39
4-3	CPCI Summary Form	40
5-1	Concepts Selected	48
5-2	Computer Program Module Dataset	52
5-3	Computer Program Component Dataset	53
5-4	Computer Program Configuration Item Dataset	54
6-1	Estimating System Structure	74
6-2	Model Logic	76
6-3	Module Breakdown	77
6-4	Models/Datasets Interrelation	81
7-1	Software Breakdown Structure for an Air Defense System	83
7-2	Allocation of the Search Modules into CPCs	85
7-3	Allocation of the Track Computer Program Requirements into CPCs	86
7-4	Allocation of the Guidance Computer Program Requirements into CPCs	87
7-5	Allocation of the Engagement Determination Computer Program Requirement into CPCs	88
7-6	Allocation of the Communications Computer Program Requirement into CPCs	89
7-7	Allocation of the Display Computer Program Requirement into CPCs	90
7-8	Allocation of the System Control Computer Program Requirement into CPCs	91
7-9	Allocation of the Utility Computer Program Requirement into CPCs	92

LIST OF TABLES

TABLE		PAGE
1-1	Software Cost Estimating Models..................	2
2-1	Software Development Effort Multipliers...........	12
2-2	Summary of COCOMO Hierarchy of Models.............	13
4-1	Software Functions................................	46
7-1	Search Modules....................................	81
7-2	Track Modules.....................................	86
7-3	Guidance Modules..................................	87
7-4	Engagement Determination Modules..................	88
7-5	Communications Modules............................	89
7-6	Display Modules...................................	90
7-7	System Control Modules............................	91
7-8	Utility Routines..................................	92

1. INTRODUCTION

This is the final report on a six month research and development effort into Cost Estimation Methodology for Command, Control, Communications, and Intelligence (C3I) System Software. The objective of the effort was to define and specify an estimating concept which could be automated for use in the Conceptual Phase of embedded software development. The approach developed was to provide improved accuracy while making maximum use of current estimation techniques and it was to be both user friendly and interactive.

During proposal preparation the software cost estimation models listed in table 1-1 were reviewed, and it was concluded that the accuracy of each depended on an input estimate of the expected number of instructions. Within the models, this estimate is then related to an average instruction productivity rate. In most cases, productivity rates were developed by regressions that explained variances in terms of factors related to programming environment, capabilities of the software analysts and programmmers, requirements for interfacing with other programs, machine constraints, and documentation needs. Since the different models are regressions of different sets of data, their basic equations require calibration to a given type of application and programming environment.

The most recently developed model reviewed was the COCOMO developed by Barry Boehm of TRW, Inc. and published in 1981 by

Table 1-1. Software Cost Estimating Models

1.0 NAVAL AIR DEVELOPMENT CENTER Model

F. Buck, et al."A Cost-By-Function Model for Avionics Computer Systems", NADC-SD-7088, March 1971

2.0 SDC Model

E.A. Nelson, T. Fleishman, "A System for Collecting & Reporting Costs in Computer Program Development", System Development Corporation, TM-3411/000/00, 11 April 1967

3.0 IBM Model

C.E. Walston, C.P. Felix, "A Method of Programming Measurement and Estimation", IBM Systems Journal, Vol 16, No. 1, pp. 54-73, 1977

4.0 GRC Model

E.N. Dodson, et al., "Advanced Cost Estimating and Synthesis Techniques for Avionics", General Research Corporation, Final Report CR-2-461, 1975

5.0 TRW Models

R.W. Wolverton, "The Cost of Developing Large Scale Software", TRW Inc., IEEE Transactions on Computer, Vol C-23, No. 6, June 1974

(COCOMO Model)
B.W. Boehm, <u>Software Engineering Economics</u>, Prentice-Hall, 1981

6.0 TECOLOTE (TEC) Model

B.C. Frederick, "A Provisional Model for Estimating Computer Program Development Costs", Tecolote Research Inc., TM-7/Rev. 1, December 1974

7.0 PUTNAM (QSM) Models

L.H. Putnam, A. Fitzsimmons, "Slim, Software Life Cycle Management Estimating Model", QSM Inc., 1978

8.0 DOTY/RADC Model

J.H. Herd, et al., "Software Cost Estimation Study I & II, Guidelines for Improved Software Cost Estimation", Doty Associates Inc., RADC-TR-77-220, February 1977

9.0 RCA Model

F. Freiman, "PRICE S Software Cost Estimating Model", RCA Price Systems, 1977

10.0 ESD Model

G.A. Bourdon, J.A. Duquette, "A Computerized Model for Estimating Software Life Cycle Costs (Model Concept)", USAF, April 1978

11.0 BOEING (BCS) Models

R.K.E. Black, et al., "BCS Software Production Data", Boeing Computer Services, March 1977

12.0 SAMSO (SAM) Model

D.L, Hansen, "Software CER Feasibility Study", Hq., SAMSO, Cost Analysis Division, December 1976

13.0 Phister Model

M. Phister, Jr., <u>Data Processing Technology and Economics</u>, Santa Monica Publishing Company & Digital Press, December 1979

14.0 MITRE Model

W. Hahn and J. Stone, Jr., "Software Transfer Cost Estimation Technique", MITRE, M70-43, July 1970

15.0 Schneider Model

V. Schneider, "Prediction of Software Effort and Project Duration -- Four New Formulas", SIGPLAN Notices, Vol 13, No. 6, June 1978

16.0 JENSEN Model

R.W. Jensen, "An Improved Macrolevel Software Development Resource Estimation Model", Procedures of ISPA Conference, April 1983
(Also implemented as the JS-1 system from Computer Electronics Inc.)

17.0 DSARC Model

B.C. DeRoze, "Embedded Computer Resources and the DSARC Process -- A Guidebook", Management Steering Committee, Embedded Computer Resources, OSD, 1977

Prentice-Hall, Inc. in a book entitled Software Engineering Economics.[1] It is based on a regression analysis of 63 data points and an extensive review of the software estimating methods developed to date. Because it incorporated this review into its factors, the COCOMO model was used as baseline for this study, and methods for programming and applying it as a user friendly interactive computer program compatible with the requirements and data available during the conceptual phase were researched. Three basic software sizing methods were reviewed to develop an approach compatible with COCOMO: analogy, computer core memory, and subjective probability. The available data to support the proposed approach was also researched and the requirements for a compatible computer program were developed.

This report presents the findings of the research and the estimating concepts and computer program development recommended.

Section 2 reports the research into methods of making the COCOMO model user-friendly. It reviews the basic models, their inputs, outputs, and assumptions, and reviews an interactive computer adaptation of COCOMO by the WANG Software Institute Graduate School (WICOMO)[2].

Section 3 reports the sizing methodology research done. It looks at analogous software sizing methods by the Aerospace and Grumman Corporations and the default library analogous estimating approach developed by the Navy "HARDMAN" project.[3,4,5] It reviews Core Memory sizing developed by Doty

Associates, subjective probability sizing using the SLIM model, and it touches on the Software Science investigations by Halstead, Elstoff, and McCabe.[6,7,8,9,10]

Section 4 reports research into software databases and current structures being used by the Air Force Electronic Systems Division (ESD) to collect C3I data. Specifically researched were the data stored in DACS (Data and Analysis Center for Software) at RADC and the Aerospace Corporation database, potential ESD project data, and the SARE (Software Acquisition Resource Expenditure) data collection methodology.[11,12,13,14]

Section 5 presents the concepts developed for a user-friendly software design/life cycle cost estimating system. It uses the COCOMO model combined with sets of generic software life cycle cost baselines called up and modified from help screens tied to estimating equations. It presents requirements for an interactive computer program to implement the developed methodology, and identifies the logic and functional modules to be programmed.

Section 6 presents the concepts developed for an interactive software design/life cycle cost estimating system that uses the COCOMO model equations and libraries of generic software C3I baseline structures called-up and modified with the use of help screens.

Section 7 presents the concept of "Sizing" libraries and gives an example as to how existing programs could be analyzed to develop generic C3I software breakdowns and hierarachy of modules of instructions that could form the basis for computer program sizing.

2. MODEL RESEARCH

2.1 COCOMO

The COnstructive COst MOdel (COCOMO) is a software development and maintenance effort estimating model that exists in a hierarchy of three increasingly detailed regressions of a database of 63 software·projects under TRW control. Project data was grouped into development mode, application type, year of development, type of computer used for development, and programming language. Regressions of delivered source instructions (DSI) against manmonths (MM) of development effort were made: one against the total number of instructions versus the mode of development; and the other, the number of instructions within a mode of development.

2.1.1 Basic Regressions

Those initial data base regressions resulted in a set of effort estimating equations called the basic COCOMO model. Those equations are the following:

Mode	Effort
Organic	$MM = 2.4(KDSI)^{1.05}$
Semidetached	$MM = 3.0(KDSI)^{1.12}$
Embedded	$MM = 3.6(KDSI)^{1.20}$

They estimate the number of manmonths required to develop a software program of a given size in terms of thousands of delivered source instructions (KDSI). The three modes of development identified in the equations are as follows:

Organic Mode -- relatively small software teams in a highly familiar, in-house environment, with extensive experience with related systems within the organization, and a thorough understanding of how the system under development will contribute to the organization's objectives. Relatively relaxed about the way the software meets its requirements and interface specifications. Generally stable development environment, with very little concurrent development of associated new hardware and operational procedures, minimal need for innovative data processing architecture or algorithms, relatively low premium on early completion of the project, and no more than 50 KDSI of new software.

Embedded -- relatively large software team operating within tight constraints to develop a product required to operate within (is embedded in) a strongly coupled complex of hardware, software, regulations, and operational procedures. A small team of analysts is used in the early stages, along with a very large team of programmers to perform detail design, coding, and unit testing in parallel. The project can be expected to expend more effort in accommodating changes and fixes; and higher costs for verification and validation and configuration management.

Semidetached Mode -- an intermediate stage between the organic and embedded modes. Accordingly, it is a mixture of the organic and embedded mode characteristic in which team members that have an intermediate level of experience with related systems, with a wide mixture of experienced and inexperienced members that have experience related to some aspects of the systems under development, but not others. The product size ranges between 50 and 300 KDSI.

The basic COCOMO model also had regressions against development time. The results of those regressions are the following schedule equations:

Mode	Schedule
Organic	$TDEV = 2.5(MM)^{0.38}$
Semidetached	$TDEV = 2.5(MM)^{0.35}$
Embedded	$TDEV = 2.5(MM)^{0.32}$

The primary variable in these equations is the number of manmonths estimated using the effort equations, and the calendar months required to develop the software (TDEV). It assumes the Rayleigh distribution function for the determination of full-time-equivalent software personnel (FSP) over the development phase:

$$FSP = MM(t/t_d^2)e^{-(t^2)/(2t_d^2)}$$

The variable t represents the month for which the FSP level is being calculated, and the quantity, t_D, represents the month at which the project achieves its peak effort.

2.1.1.1 Adaptation of Software

The basic COCOMO Model estimates the development effort and schedule time for the adaptation of existing software in terms of an equivalent number of new delivered source instructions (EDSI), which is used in the place of DSI in the COCOMO estimating relationships. The equations for calculating EDSI involve an intermediate quantity, the adaptation adjustment factor (AAF).

$$\text{EDSI} = (\text{ADSI})(\text{AAF}/100)$$

where, $\text{AAF} = 0.40(\text{DM}) + 0.30(\text{CM}) + 0.30(\text{IM})$

and ADSI = Adapted DSI

DM = Percent design modified

CM = Percent code modified

IM = Percent of integration required for modified software

The coefficients in the AAF were determined from the average fractions of effort devoted to design, code, and integration-and-test in the COCOMO data base.

2.1.1.2 Software Maintenance

COCOMO uses an estimated Annual Change Traffic (ACT) factor to estimate annual maintenance manhours $(\text{MM})_{\text{AM}}$ for a software program. ACT is the fraction of the software's source instructions expected to undergo change during a typical year, either through addition or modification.

$$(\text{MM})_{\text{AM}} = (1.0)(\text{ACT})(\text{MM})_{\text{DEV}}$$

Another alternate factor for estimating overall life-cycle maintenance manhours $(\text{MM})_{\text{M}}$, from acceptance test through phaseout is the maintenance/development manhour ratio (M/D).

$$(MM)_M = (M/D)(MM)_{DEV}$$

A third alternative is an estimate of the thousands of source instructions maintained per full-time software person, and the number of maintenance personnel $(FSP)_M$ required to support a given size development $(KDSI)_{DEV}$.

$$(MM)_{AM} = 12(FSP)_M$$

where,

$$(FSP)_M = (KDSI)/(KDSI/FSP)_M$$

The software maintenance data in the COCOMO data base reflect a range of cards per person (KDSI/FSP) from 3.2 to 132 with a median of 25, a maintenance productivity (DSI/MM) from 36 to 1238 with a median of 164, and an Annual Change Traffic (ACT) from 0.01 to 0.4 with a median of 0.08.

2.1.1.3 Computer Time

COCOMO adds the cost of computer time used in development of software and the cost of clerical personnel to the effort cost estimates. Computer time is estimated from historical characteristics of projects wherein computer hours per development manmonth have been determined for maxi, midi, and mini type computers. Small to median size timeshared developments used 0.2 to 1.5 hours computer time per development manmonth; large or batch application developments used 3 hours per manmonth; and real-time developments used between 3 to 18 hours per manmonth. The smaller the computer used the larger the number of hours.

2.1.1.4 Clerical Effort

Basic COCOMO estimates cover the cost of professional personnel and paraprofessionals such as program librarians, but not clerical effort. Three to four percent of the basic manpower estimate must be added to cover the clerical effort.

2.1.2 Intermediate Models

COCOMO regressions are further refined in an "intermediate" model to reflect added sets of cost driver attributes:

o Product Attributes

RELY Required Software Reliabiltiy

DATA Data Base Size

CPLX Product Complexity

o Computer Attributes

TIME Execution Time Constraint

STOR Main Storage Constraint

VIRT Virtual Machine Volatility

TURN Computer Turnaround Time

o Personnel Attributes

ACAP Analyst Capability

AEXP Applications Experience

PCAP Programmer Capability

VEXP Virtual Machine Experience

LEXP Programming Language Experience

o Project Attributes

MODP Modern Programming Practices

TOOL Use of Software Tools

SCED Required Development Schedule

The intermediate model equations are as follows:

Development Mode	Nominal Effort Equation
Organic	$MM_{NOM} = 3.2(KDSI)^{1.05}$
Semidetached	$MM_{NOM} = 3.0(KDSI)^{1.12}$
Embedded	$MM_{NOM} = 2.8(KDSI)^{1.20}$

The effort multipliers related to the intermediate model are abstracted in table 2-1. Except for SCED (Required Development Schedule), RELY (Required Software Reliability), and MODP (Modern Programming Practices) the effort multipliers can be applied to the maintenance effort estimate as well as development. SCED is only a factor during development, not maintenance, and RELY and MODP have different multipliers for maintenance effort estimating. The same computer time and clerical effort relationships are used.

Table 2-1. Software Development Effort Multipliers[1]

Cost Driver	Multiplier Range		
	Low	Nominal	High
Product Attributes			
Required software reliability	.75	1.00	1.40
Data base size		1.00	1.16
Product complexity	.70	1.00	1.65
Computer Attributes			
Execution time constraint		1.00	1.66
Main storage constraint		1.00	1.56
Virtual machine volatility		1.00	1.30
Computer turnaround time		1.00	1.15
Personnel Attributes			
Analyst capability	1.46	1.00	.71
Applications experience	1.29	1.00	.82
Programmer capability	1.42	1.00	.70
Virtual machine experience	1.21	1.00	
Programming language experience	1.14	1.00	
Project Attributes			
Use of modern programming practices	1.24	1.00	.82
Use of software tools	1.24	1.00	.83
Required development schedule	1.23	1.00	1.10

2.1.3 Detailed Model

The final codification of the COCOMO regressions was the development of separate effort multipliers for each major development phase. These multipliers are applied at a three level hierarchical decomposition of the software product whose cost is to be estimated. The lowest level, the module level effort equations, is estimated by the intermediate model equation cost drivers that vary at the lowest level. They are: the module's complexity and adaptation from existing software, programmers' capability and experience with the language, and the virtual machine on which the software is built. The subsystem level effort is modified by the remainder of the cost drivers (storage constraint, analysts capability, tools, schedule, etc.) which tend to vary from subsystem to subsystem, but which tend to be the same for all the modules within a system.

Two work sheets are provided for input use, CLEF (Component Level Estimating Form) and SHEF (Software Hierarchy Estimating Form), found in the Prentice-Hall book.[1] Table 2-2 summarizes the similarities and differences among the three levels of the COCOMO hierarchy of models (Basic, Intermediate, and Detailed).

Table 2-2. Summary of COCOMO Hierarchy of Models[1]

	COCOMO Level		
Estimate	Basic	Intermediate	Detailed
Development effort MMDEV	mode, KDSI	mode, KDSI, 15 cost drivers	mode, KDSI, 15 cost drivers by phase
Development schedule	mode, MMDEV	Same as for Basic	Same as for Basic
Maintenance effort	MMDEV, ACT)	MMDEV, ACT, 15 cost drivers and 2 maintenance drivers	Same as for Intermediate
Product hierarchy	Entire system	System/components CLEF form and procedures	System/subsystem/module SHEF form and procedures
Phase distribution of effort	mode, KDSI	Same as for Basic	mode, KDSI, 15 cost drivers by phase
Phase distribution of schedule	mode, KDSI	Same as for Basic	Basic schedule distribution Detailed effort distribution
Activity distribution	mode, KDSI	Same as for Basic	Same as for Basic
Requirements phase effort percentage	mode, KDSI	Same as for Basic	mode, KDSI, 15 cost drivers by phase

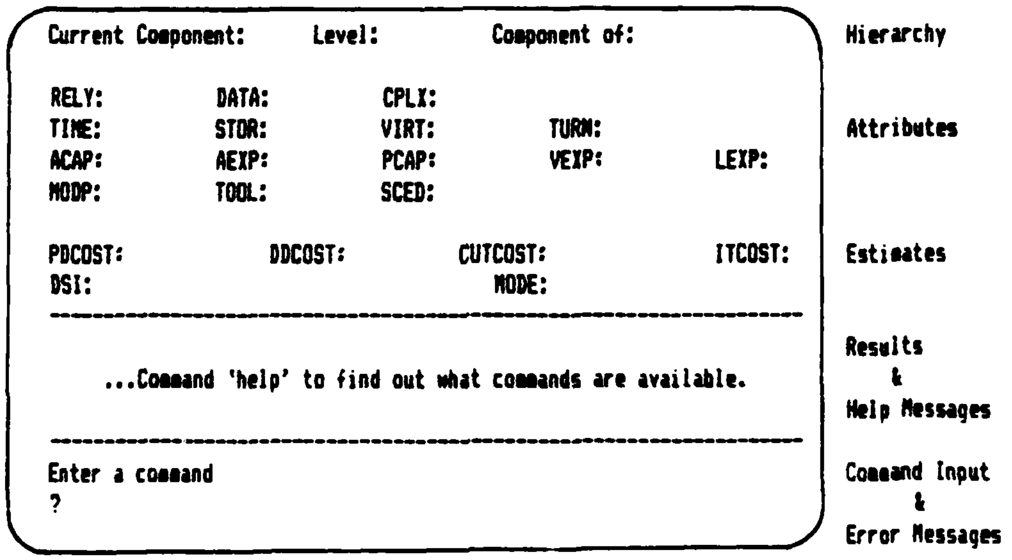
2.2 WICOMO

WICOMO (<u>W</u>ang <u>I</u>nstitute <u>Co</u>st <u>Mo</u>del) is the Wang Institute's computerized implementation of the COCOMO model[2]. It was developed in the Winter 1982 Project I course at the Wang Institute under the supervision of Dr. James P. Bouhana.

WICOMO is user friendly and alleviates much of the problem of having so many inputs needed to accomplish an estimate by providing a default baseline and definition "help" screens. It generalizes COCOMO's system-subsystem-module hierarchy such that it can be extended to any number of levels. It also generalizes the COCOMO cost driver attributes to any level of the software hierarchy defined. Values specified at higher levels become the default values for lower level components. Thus, an attribute which will be constant for the entire system need be specified only once at the topmost level of the hierarchy, while attributes which vary at the lowest level can be specified for each such component individually.

Cost attribute values are restricted to standard rating levels. Interpolation can be accomplished only by modification of the effort multiplier tables. However, these, along with all other numerical values associated with COCOMO, are obtained from an external file. This allows easy calibration to fit the experience of a specific organization. The interactive approach used is illustrated with the basic WICOMO display which is shown in figure 2-1. It contains the fundamental elements of the COCOMO effort estimating relationships and an identification of the level of software hierarchy being estimated. The upper part of the screen is used to display the values of all attributes of the component. The lower part of the screen is used for displaying results and help messages. The bottom two lines are used for command input and error messages respectively.

After the development mode and estimated number of delivered source instructions are entered, an estimate of development cost can be made. Unless an estimate for each attribute is also entered, the development cost estimate made would be with "nominal" defaults for each of the attributes. Any of the attributes can be changed and there are "help" screen to aid in their estimation.



```
Current Component:      Level:          Component of:

RELY:           DATA:           CPLX:
TIME:           STOR:           VIRT:           TURN:
ACAP:           AEXP:           PCAP:           VEXP:           LEXP:
MODP:           TOOL:           SCED:

PDCOST:             DDCOST:          CUTCOST:                ITCOST:
DSI:                                    MODE:
--------------------------------------------------------------------

     ...Command 'help' to find out what commands are available.

--------------------------------------------------------------------
Enter a command
?
```

Figure 2-1. WICOMO Interactive Screen Format

WICOMO decomposes software systems into lower levels by estimating source instruction counts at succeeding lower levels. Cost driver attributes are inherited by each lower level and instruction counts are always summed to the higher level. Changes are automatically propagated.

Three basic reports are available from WICOMO: RESULTS, SUMMARY, and SCHEDULE. The "summary" and "schedule" reports are only developed at the system level. The "schedule" report presents a month-by-month schedule of man month and dollar expenditures.

3. SIZING RESEARCH

All computer program sizing is based on some type of functional decomposition of requirements. Decomposition starts early in a development and continues until each requirement is decomposed to a level low enough to be allocated to hardware, software, or a procedure. Once requirements have been allocated between hardware and software, software modules are identified, named, and sized in source lines of code.

3.1 Analogous Sizing

In analogous sizing instruction countsfrom similar software programs developed in the past are used to help in the actual module sizing activity. Research was conducted into methodologies for implementing analogous sizing in a user friendly interactive computer program. Three activities were investigated: the Aerospace Corporation's Guidlines, Grumman's Software Cost Estimating Model, and the Navy's Hardman Project Life Cycle Cost Model.

3.1.1 Aerospace Method

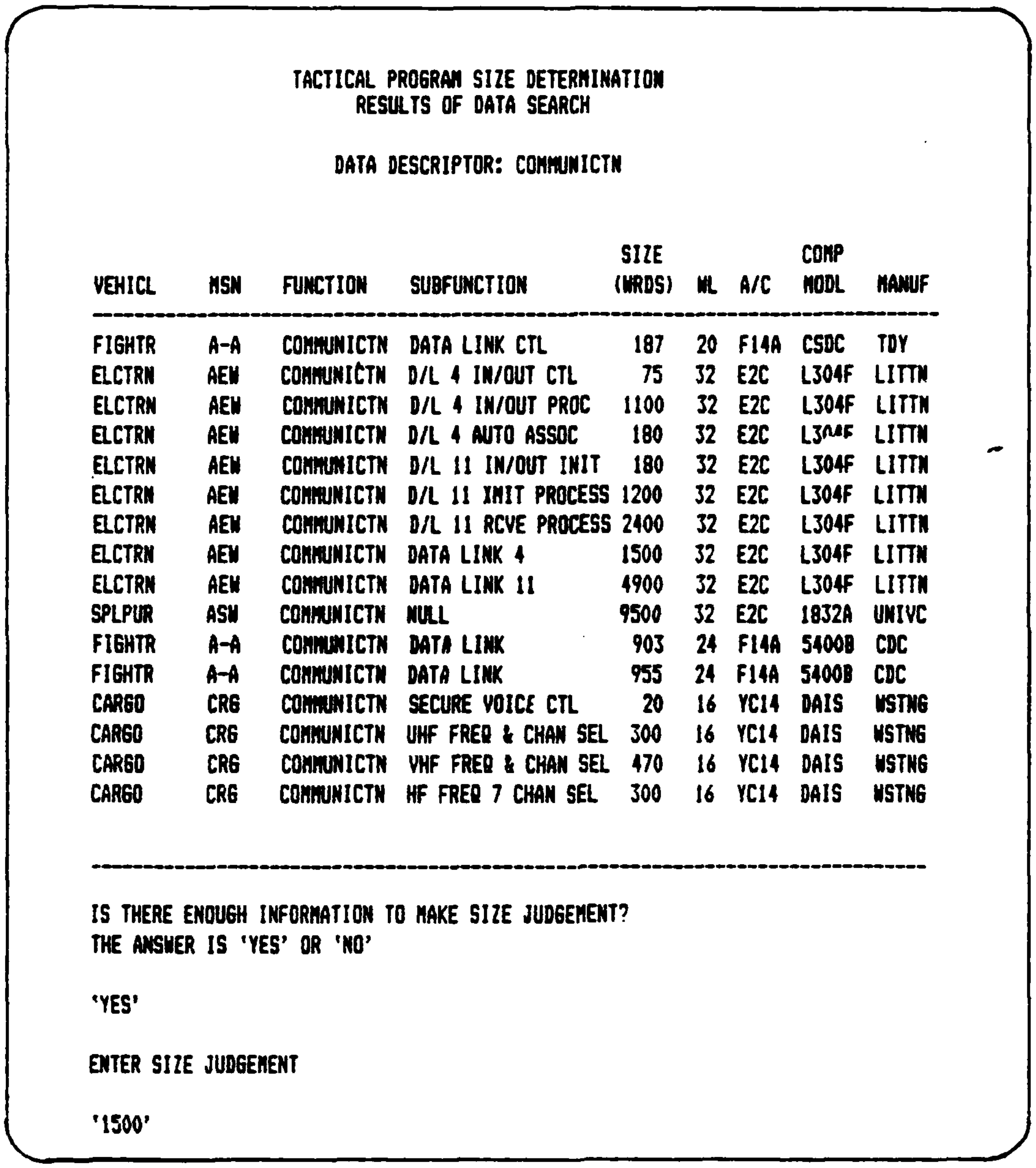
The Aerospace method does analogous sizing methodology in two basic steps. First, a software work breakdown structure is developed, then instructions for the lowest level items in the breakdown are estimated by engineering judgement. The lowest level of the breakdown is to the functional level. Analogous data is grouped by ranges of instruction for different types of functions. Engineering judgements are made with respect to where, in the range of instructions, the program of interest lies. Judgements are made based on three considerations:

> Complexity - Items to consider include required accuracy or precision of the outputs, the amount of autonomy in the function, and the survivablity of the application.
>
> Application - Consideration of the sameness of the application of the software function compared with the applications in the database.
>
> Extensiveness - The extensiveness of the requirements contained in the function to be estimated is compared with those in the database (i.e., the number of data links, the number of secure data lines, the number and types of interfaces, etc..)

3.1.2 Grumman Method

The Grumman approach is similar to Aerospace's except it is done on the computer in a cost estimating model called "SOFCOST". It is executed as an interactive computer program in which the estimator is coached in deriving a software work breakdown structure (SWBS) and estimating programs size by judgements based on a stored database of historical SWBS size data.

The estimator, through an interactive terminal session, describes the system requirements such that a SWBS is established and displayed. The highest software level of this structure is the computer program configuration item. The next level is the "category" of software and the lowest level is the function within a category. For each of the functions established in the desired work breakdown structure, a functional size data base is searched. After viewing the displayed sizes the estimator compares this output with his knowledge of the functional requirement being estimated. A size judgement is then made and entered into the model. Figure 3-1, from the IEEE paper, shows an example of the type of display of size information given.



```
               TACTICAL PROGRAM SIZE DETERMINATION
                    RESULTS OF DATA SEARCH

                  DATA DESCRIPTOR: COMMUNICTN


                                                SIZE              COMP
VEHICL   MSN   FUNCTION    SUBFUNCTION          (WRDS)  WL  A/C   MODL   MANUF
-------------------------------------------------------------------------------
FIGHTR   A-A   COMMUNICTN  DATA LINK CTL          187   20  F14A  CSDC   TDY
ELCTRN   AEW   COMMUNICTN  D/L 4 IN/OUT CTL        75   32  E2C   L304F  LITTN
ELCTRN   AEW   COMMUNICTN  D/L 4 IN/OUT PROC     1100   32  E2C   L304F  LITTN
ELCTRN   AEW   COMMUNICTN  D/L 4 AUTO ASSOC       180   32  E2C   L304F  LITTN
ELCTRN   AEW   COMMUNICTN  D/L 11 IN/OUT INIT     180   32  E2C   L304F  LITTN
ELCTRN   AEW   COMMUNICTN  D/L 11 XMIT PROCESS   1200   32  E2C   L304F  LITTN
ELCTRN   AEW   COMMUNICTN  D/L 11 RCVE PROCESS   2400   32  E2C   L304F  LITTN
ELCTRN   AEW   COMMUNICTN  DATA LINK 4           1500   32  E2C   L304F  LITTN
ELCTRN   AEW   COMMUNICTN  DATA LINK 11          4900   32  E2C   L304F  LITTN
SPLPUR   ASW   COMMUNICTN  NULL                  9500   32  E2C   1832A  UNIVC
FIGHTR   A-A   COMMUNICTN  DATA LINK              903   24  F14A  5400B  CDC
FIGHTR   A-A   COMMUNICTN  DATA LINK              955   24  F14A  5400B  CDC
CARGO    CRG   COMMUNICTN  SECURE VOICE CTL        20   16  YC14  DAIS   WSTNG
CARGO    CRG   COMMUNICTN  UHF FREQ & CHAN SEL    300   16  YC14  DAIS   WSTNG
CARGO    CRG   COMMUNICTN  VHF FREQ & CHAN SEL    470   16  YC14  DAIS   WSTNG
CARGO    CRG   COMMUNICTN  HF FREQ 7 CHAN SEL     300   16  YC14  DAIS   WSTNG


-------------------------------------------------------------------------------

IS THERE ENOUGH INFORMATION TO MAKE SIZE JUDGEMENT?
THE ANSWER IS 'YES' OR 'NO'

'YES'

ENTER SIZE JUDGEMENT

'1500'
```

Figure 3-1. "SOFCOST" Search Display Example[13]

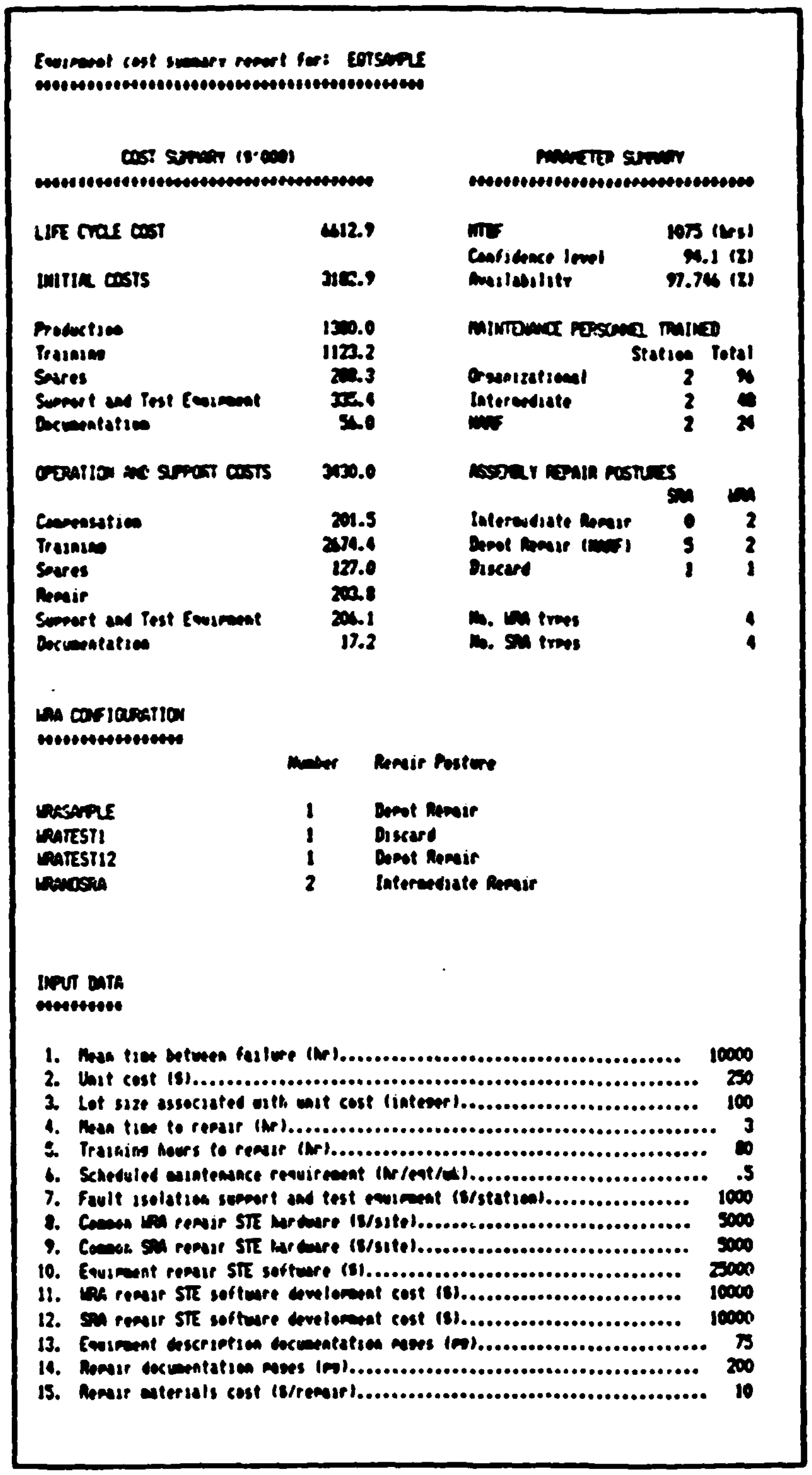
3.1.3 HARDMAN Model

The Navy's HARDMAN Project Life Cycle Cost Model is not a software cost or sizing model, but its spreadsheet and library features are worth considering for analogous estimating adaptation.

The estimating system consists of four linked programs that combine to estimate life cycle cost of equipments, assemblies, and subassemblies: an environment data set program, an equipment design/cost model, a Weapon Removable Assembly (WRA) design/cost model, and a Shop Removable Assembly (SRA) design/cost model. Each model is similar in structure. Each allows manipulation of input data files that describe the design of an equipment, assembly, or subassembly, and each computes the life cycle cost of a proposed design at its assigned level. In each case when a data set is created, it becomes part of a permanent data library stored on disk. Equipments are designed by entering equipment-level parameters and choosing from the library of stored WRAs. WRAs are designed by entering WRA level parameters and choosing from the library of stored SRAs. The SRA is the generic building block.

The Environment Data Set program requires both environmental and cost factors that are common to the three levels of equipment and do not depend on the type of equipment, nor equipment design. These data are common input to the other models. Sets of these data can be stored in the data library and designated for use with a given design.

The Equipment Design/Cost program allows the creation of a library of alternate equipment designs and the estimation of the life cycle cost of each alternative. The WRA Design/Cost program allows the creation of a library of alternate WRA designs and the estimation of the life cycle cost of each alternative. The SRA Design/Cost program allows the creation of a library of SRA designs and the estimation of the life cycle cost of each alternative. Figure 3-2 illustrates the information displayed from the library for an equipment. Similar displays are stored for WRAs and SRAs.



```
Equipment cost summary report for:  EQTSAMPLE
********************************************


         COST SUMMARY ($'000)                          PARAMETER SUMMARY
****************************************       ********************************

LIFE CYCLE COST                  6612.9        MTBF                  1075 (hrs)
                                               Confidence level        94.1 (%)
INITIAL COSTS                    3182.9        Availability          97.746 (%)

Production                       1380.0        MAINTENANCE PERSONNEL TRAINED
Training                         1123.2                            Station  Total
Spares                            288.3        Organizational            2     96
Support and Test Equipment        335.4        Intermediate              2     48
Documentation                      56.0        NARF                      2     24

OPERATION AND SUPPORT COSTS      3430.0        ASSEMBLY REPAIR POSTURES
                                                                       SRA    WRA
Compensation                      201.5        Intermediate Repair       0      2
Training                         2674.4        Depot Repair (NARF)       5      2
Spares                            127.0        Discard                   1      1
Repair                            203.8
Support and Test Equipment        206.1        No. WRA types                    4
Documentation                      17.2        No. SRA types                    4


WRA CONFIGURATION
*****************
                                Number     Repair Posture

WRASAMPLE                          1       Depot Repair
WRATEST1                           1       Discard
WRATEST12                          1       Depot Repair
WRANOSRA                           2       Intermediate Repair



INPUT DATA
**********

 1.  Mean time between failure (hr).....................................  10000
 2.  Unit cost ($)...........................................................  250
 3.  Lot size associated with unit cost (integer)..........................  100
 4.  Mean time to repair (hr)..................................................  3
 5.  Training hours to repair (hr)...........................................  80
 6.  Scheduled maintenance requirement (hr/eqt/wk)...........................  .5
 7.  Fault isolation support and test equipment ($/station)................  1000
 8.  Common WRA repair STE hardware ($/site)...............................  5000
 9.  Common SRA repair STE hardware ($/site)...............................  5000
10.  Equipment repair STE software ($).....................................  25000
11.  WRA repair STE software development cost ($).........................  10000
12.  SRA repair STE software development cost ($).........................  10000
13.  Equipment description documentation pages (pg)...........................  75
14.  Repair documentation pages (pg).........................................  200
15.  Repair materials cost ($/repair)..........................................  10
```

14

Figure 3-2. Equipment Library File

3.2 Core Requirements Sizing

Doty Associates developed an algorithm for software sizing based on the number of functions to be programmed and the size of the memory of the computer being programmed. The algorithm was developed by a multivariate regression of data obtained from a study by the John Hopkins University Applied Physics Laboratory. That algorithm is as follows:

$$M = [(N_F^{0.337})(W_S^{0.147})/(t_C^{0.770})]e^{0.177 + K}$$

where,

M = Memory size in thousands of words of object code

N_F = Number of major functions to be performed by the software

W_S = Word size in bits

t_C = Cycle time of processor in microseconds

K = A constant dependent on application

where K equals:

2.573	for signal processing
2.727	for missile fire control
2.781	for interfacing
3.412	for communication
3.565	for navigation
4.046	for command and control
4.451	for weapon fire control

The variable, N_F, is defined as being functions such as communications, target tracking, target identification, navigation, system monitoring, display, steering, parameter measurement, tuning, target data entry, timing sequence control, etc.. W_S and t_C are defined by the CPU of the computer system planned to be used. By assuming that average core utilization is approximately 80 percent, the Doty algorithm can be used for estimating system size. In order to do this a HOL code to object code and word size must be made. A summary of object code/source instruction expansion ratios are given in the Boehm book.

3.3 PERT Sizing

The SLIM model uses what has come to be known as the PERT sizing method for software sizing. According to a paper presented by Dean,[7] the SLIM model uses its EDITOR model to determine this. It requests three inputs:

A, the smallest possible number of source statements

M, the most likely number of source statements

B, the largest possible number of source statements.

It then uses each of these inputs to get an expected number of lines of code (E_i) by using the formula

$$E_i = (A + 4M + B)/6$$

It computes the standard deviation of each input by the relationship:

$$\sigma_i = (B - A)/6$$

The probability of a given number of lines of code is estimated by adding and subtracting the required number of standard deviations.

3.4 Software Science

In 1977 M. H. Halstead published a theory of software complexity called "Software Science".[8] That theory contained a measure of computer program size in terms of program operators and operands. Operators include arithmetic operators (e.g., +, −, *, /), logical operators (e.g., greater than, equal to), and keywords (e.g., FORTRAN DO, COBOL PERFORM), and delimitors. Operands include constants and variables.

n_1 = number of distinct operators in program
n_2 = number of distinct operands in program
N_1 = total number of operators in program
N_2 = total number of operands in program

The length of a program, is simply

$$N = N_1 + N_2$$

The vocabulary, n, of a program is simply

$$n = n_1 + n_2$$

Elshoff at General Motors Research Laboratories calculated estimated length, N as follows

$$N = n_1 \log_2 n_1 + n_2 \log_2 n_2$$

In addition, Elshoff found that the estimated length, N_1, more closely equated the actual length, N_1, for well-structured programs.[9]

There have been studies that correlate N to the number of source instructions required. They were not pursued during this contract; however, as will be seen later, information on operators and operands are available in the NASA/SEL database.

Along this same line, McCabe has suggested a graph-theoretic complexity measure of computer program complexity called the "cyclomatic number".[10] For structured programs, cyclomatic complexity can be calculated by simply counting the number of compares:

cyclomatic complexity = compares + 1

Complexity evaluation is applied at the module level in a program. It is used to control the size of a program and hence its understandability from a maintenance standpoint.

4. DATA RESEARCH

A search was made of the type of data that would be available for analogous software sizing and cost model verification and validation. The search was made through the DACS, the Data and Analysis Center for Software, operated by IIT Research Institute under contract to RADC. As a result, an analysis was made of the DACS Software Life-cycle Emperical Database (SLED), the Areospace Corporation's Database, and the Electronics Systems Division programs on which machine data was collected by Doty Associates in their 1980 sizing studies.[11,3,6]

4.1 RADC Data

The DACS has acquired seven sets of data from various sources and maintains this data in the Software Life Cycle Empirical Database. The seven sets of data are the following:

1) The DACS Productivity Dataset
2) The Reliability Dataset
3) The NASA/SEL Life Cycle Dataset
4) The Verification & Validation (V&V) Dataset
5) The ARF Error Dataset
6) The Baseline Software Dataset
7) The Operations and Maintenance (O&M) Dataset

Several of these datasets should be useful to analogous sizing.

4.1.1 DACS Productivity Dataset

This dataset consists of summary data on roughly 400 software projects and was compiled by Richard Nelson of RADC. The data was collected from open literature and private sources in industry and government and represents software development projects dating from the early 1960's through the mid 1970's. The software applications range from avionics and space-flight command and control functions and radar system support, to off-the-shelf software packages, communications software, and management information systems. Most of the projects represent DOD or other government applications.

The dataset identifies eight parameters and several derived factors for the different projects it contains; however, not all parameters are available on each project. The eight parameters identified are the following:

1) Project Identification

2) Project Size

3) Project Effort

4) Project Duration

5) Source Code Language

6) Errors

7) Documentation

8) Implementation

Project size is the number of lines of source code. Source lines are 80 character source records (assembly language) provided as input to a language processor. Where the size of the code has been given in computer words, an arbitrary conversion to DSLOC was made dividing the computer words by two for high order language DSLOC. Errors are the number of formally recorded Software Problem Reports (SPR). Documentation is delivered pages of documentation including program listings, flow charts, operating procedures, maintenance procedures, and other descriptive material. Implementation is the techniques, such as structured coding, top down design and programming, chief programmer teams, code reviews or inspections, and librarian or program support library.

The derived factors are the following:

1) Productivity (DSLOC/TMM)

2) Average Number of Personnel (TMM/TM)

3) Error Rate (ERRS/DSLOC)

4) Error Rate (temperal)(ERRS/TMM)

5) Documentation Rate (DOC/DSLOC)

4.1.2 Reliability Dataset

This dataset consists of software failure data compiled by John Musa of Bell Telephone Laboratories. The data was collected throughout the mid 1970's and represents projects of a variety of applications including real time command and control, word processing, commercial and military. For each software failure in the dataset the following items are recorded:

1) Project Identification
2) Failure Number
3) Failure Interval
4) Day of Failure

4.1.3 NASA/SEL Dataset

The NASA Software Engineering Laboratory (SEL) at Goddard Space Flight Center collects extensive data on software developed by their Systems Development Section. Projects represented in the dataset span the functions of attitude determination, attitude control, maneuver planning, orbit adjustment, and general mission analysis support systems. The data is stored in eleven files. These files are the following:

1) Encoding Dictionary File
2) Estimated Statistics File
3) Header File
4) Change Report File
5) Component Status Report File
6) Component Summary File - part 1
7) Component Summary File - Part 2
8) Resource Summary File

9) Run Analysis File

10) Component Information File

11) Growth History File.

The Encoding Dictionary file defines the code used in the other files. The Estimated Statistics file summarizes actual, not estimated, size, effort, and source environment data on a project. The Header file provides schedule dates for life cycle milestones of the project. The Change Report file records effort and type of changes. The Component Status Report file records the hours spent each week during development on design, code, and test. The Component Summary file summarizes, for each component of the program, complexity, application, size, schedule, effort to develop, and language. The Resource Summary file records the consumption of resources for a specified time period, including manpower, computer, and support services. The Run Analysis file records the objectives and results of each computer job submitted and whether the run was interactive or batch. The Component Information file provides information on software science metrics, and instruction mix parameters. This information is obtained from "Source Analyzer" programs. The Growth History file is a weekly accumulation of source lines written, modules, and changes.

Of particular interest in the NASA/SEL Dataset is the classification of components as combinations of the following types of functional software:

1) I/O Processing
2) Algorithmic
3) Logic Control
4) System Related
5) Data/COMMON Blocks
6) Other

and the following software module details:

1) Number of Executable Statements
2) Number of Lines with Comments
3) Number of Comment Lines
4) Number of Unique Operators
5) Number of Unique Operands
6) Total Number of Operators
7) Total Number of Operands
8) Number of I/O Variables from Module
9) Number of Decisions (McCabe's Measure)
10) Number of FUNCTION References
11) Number of I/O Statements
12) Number of Assignment Statements
13) Number of CALL Statements
14) Number of FORMAT Statements

4.1.4 Verification and Validation Dataset

This dataset contains data collected during the independent verification and validation (V&V) of five software projects. Although the specific projects are not identified an overall classification is made as to whether or not a project is C3I or not. HOL and Assembly language lines of code are given and the programming practices used identified. The primary purpose of the dataset is to record the type of errors which can occur during V&V activities, not software sizing. The general size of the projects reported are from 14,000 to 52,000 lines of code.

4.1.5 Operations & Maintenance Dataset

This dataset is data collected against the PAVE Phased Array Warning Systems (PAWS). The PAVE PAWS is an over-the-horizon radar system in operation at Otis Air Force Base and Beale Air Force Base. The data collected is maintained in seven files:

1) Maintenance Activity File
2) CPCG Description File
3) CPCG Status File
4) Segment Change History File
5) Change History File
6) Discrepancy Report History File
7) Personnel Experience Profile

The Computer Program Configuration Group (CPCG) Description and Status files may be of use in sizing. The CPCG is a subgroup of computer program configuration items. The CPCG Description file contains data providing information on the characteristics of the PAVE/PAWS software at the CPCG level, including size in source lines and words of machine code, environmental factors, and development constraints. The CPCG status file contains information on the size of the CPCG and its revision identification, along with change information. Much of the information in the file is compatible with the

the COCOMO model requirements.

4.2 Aerospace Data

The Aerospace Corporation developed a software sizing data base in 1983.[3] The data base contains data concerning software size versus software function at the subsystem and component level. They are directly used in analogous sizing. Some of the data in the data base is at the CPCI level, some at the CPC level, and still others at the module level. The data includes information on the software function, the size in lines of code, the system, the type of application, the development status, the language in which the software was written, the complexity of the technical requirements of that function, the computer on which the software was hosted, and the word size of that computer.

A five level software work breakdown structure is used to correlate functions to applications, to environments, to platforms, to system. An example of the software work breakdown structure is shown in figure 4-1. The application level is equivalent to a CPCI, the function level is equivalent to a CPC or a module.

The data base contains ranges for certain software functions. These ranges were based on engineering judgement as to what constituted a similar function and what requirements were included. Typical standard software functions isolated in the data base are the following:

Attitude determination and control

Automatic gain control

Attitude maneuver

Antenna pointing

Command generation

Command guidance system

Commanding

Command and control (C2)

Command, control and communications (C3)

Diagnostics

Data base routine

Data reduction

Display management

•

•

•

Etc.

		Project Name					SYSTEM
Flight				Ground			PLATFORM
Avionics	Unmanned Space	Manned Space	Naval	Fixed	Mobile	Remote	ENVIRONMENT
Spacecraft	Payload	Support	Mission Planning	Data Reduction	Command & Control		APPLICATION
Attitude Manuever	Antenna Printing	Utilities	Housekeeping	Telemetry Processing	Attitude Determination		FUNCTION

Figure 4-1. Software Work Breakdown Structure for Aerospace Data

4.3 ESD Project Data

4.3.1 Projects

The addendum to the ESD "Handbook of Procedure for Estimating Computer System Sizing and Timing Parameters" contains the types of data that can be used in the development of computer system analogies for for C3I core memory sizing.[6] It contains a listing of typical ESD C3 major projects, and associated listing of generalized computer equipment specifications for some ESD systems. Typical of the sample projects identified are the following:

- o Air Force Satellite Communications System 1205
- o Air Force World Wide Military Command and Control System
- o Cobra Dane 633A
- o Combat Grande
- o Combat Theater Communications 478T
- o CONUS Over-the-Horizon Backscatter Radar 414L
- o E-3A Airborne Warning and Control System (AWACS) 411L
- o E-4 Airborne Command Post 481B
- o Joint Surveillance System 968H
- o Joint Tactical Information Distribution System 634B
- o NORAD Cheyene Mountain Complex Improvements 427M
- o PAVE PAWS 2054
- o SAC Digital Information Network (SACDIN) 1136T
- o Tactical Air Control System Improvements (TACSI) 4856

Typical of the computer equipment specifications identified are the following:

- Control Data Corporation (CDC) Cyber 74
- Control Data Corporation (CDC) Cyber 174-12
- Control Data Corporation (CDC) System 17
- Control Data Corporation (CDC) AN/UYK-25 MP60
- Data General NOVA 840
- Data General NOVA 1220
- Digital Equipment Corporation (DEC) PDP 11/05
- Digital Equipment Corporation (DEC) PDP 11/10 [and 11/40]
- Honeywell H-716
- Honeywell H-6050 and 6060
- Honeywell H-6080
- IBM 370/155
- Intel 80
- Raytheon RDS-500
- Rolm 1603
- Texas Instruments TI-980A
- UNIVAC AN/UYK-7
- UNIVAC AN/UYK-20 (V-1600)
- UNIVAC AN/UYK-1108
- UNIVAC AN/UYK-1110
- UNIVAC AN/UYK-1616

The report identifies the primary functions and characteristics for each computer used by each system. For example, it provides detailed information on the following computer characteristics:

Data Format

Main Storage

Central Processor

Input/Output Control

Peripheral Equipment

4.3.2 SARE Data Collection Methodology

The Software Acquisisiton Resource Expenditure (SARE) data collection methodology is being developed by the MITRE Corporation under the direction of the Electronics Systems Division (ESD) of the Air Force.[12] It will be used by ESD to collect cost (dollars and hours) and schedule data on software developments and correlating technical characteristics. It establishes software-related Work Breakdown Structure elements for consistent cost data collection across programs, and it provides a data item description (DID) for software cost data collection that can be referenced in the contract data requirements list (CDRL) of the contract.

A draft military standard provides definitions for prime mission software decomposition:

"Prime mission software (software sytem). The aggregate of all computer programs and databases that operate as part of the defense system. This includes applications software developed specifically to provide a prime mission function of the defense systems and support software, such as off-the-shelf operating systems, data base management systems, on-line diagnostics, etc., which execute in the target computer(s) during any mode of system operation....The prime mission software may be partitioned directly into computer program configuration items or it may be partitioned into software subsystems which are in turn partitioned into computer program configuratin items...

Software subsystem. A subdivision of the software system which operates as an integral whole and provides a major function of the system. A software subsystem is comprised of two or more computer program configuration items...

Computer program configuration item (CPCI). An aggregation of software, or any of its discrete portions which satisfies an end use function and has been designated by the government for configuration management...

Computer program component (CPC). A functionally or logically distinct part of a CPCI distinguished for convenience in designing and specifying a complex CPCI as an assembly of subordinate elements..."[15]

Requirements for extended CPCI contract work breakdown structure elements were given in the draft MIL-STD. Figure 4-2 illustrates the specified breakdown of a CPCI.

The draft MIL-STD is used in conjunction with the draft DID. The draft DID references the Boehm book Software Engineering Economics, and the "NASA/SEL Data Collection Forms". This makes the proposed data collection compatible with the COCOMO model. There are Project Summary and CPCI Summary Forms provided with the DID. The Project Summary is six pages

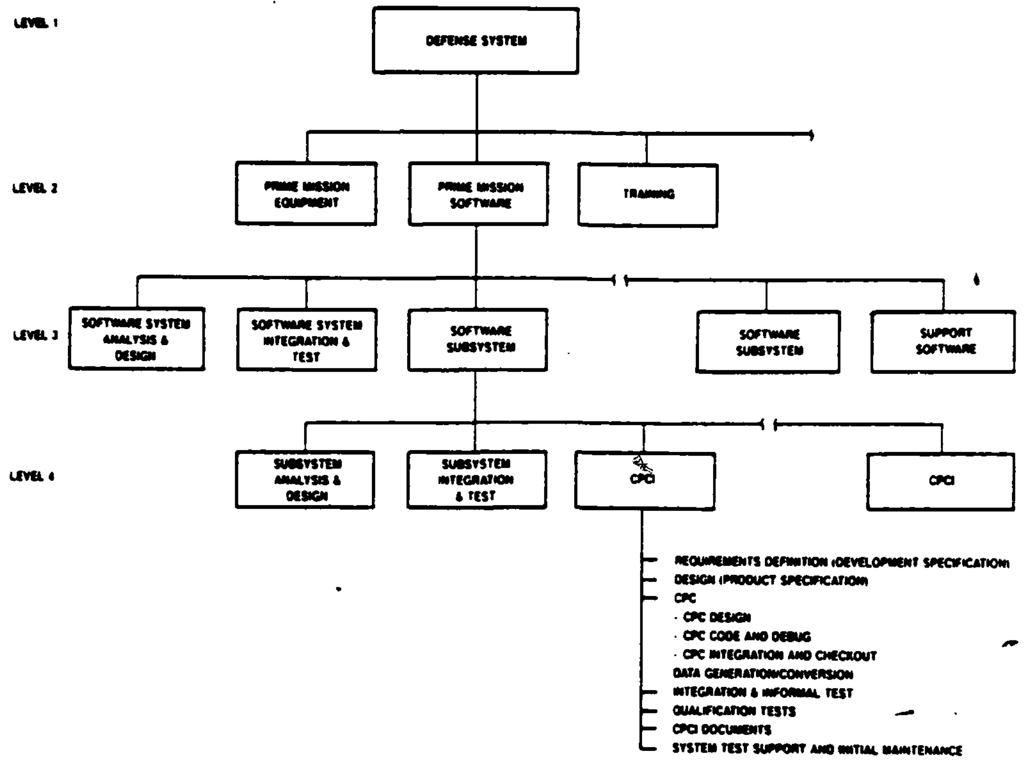
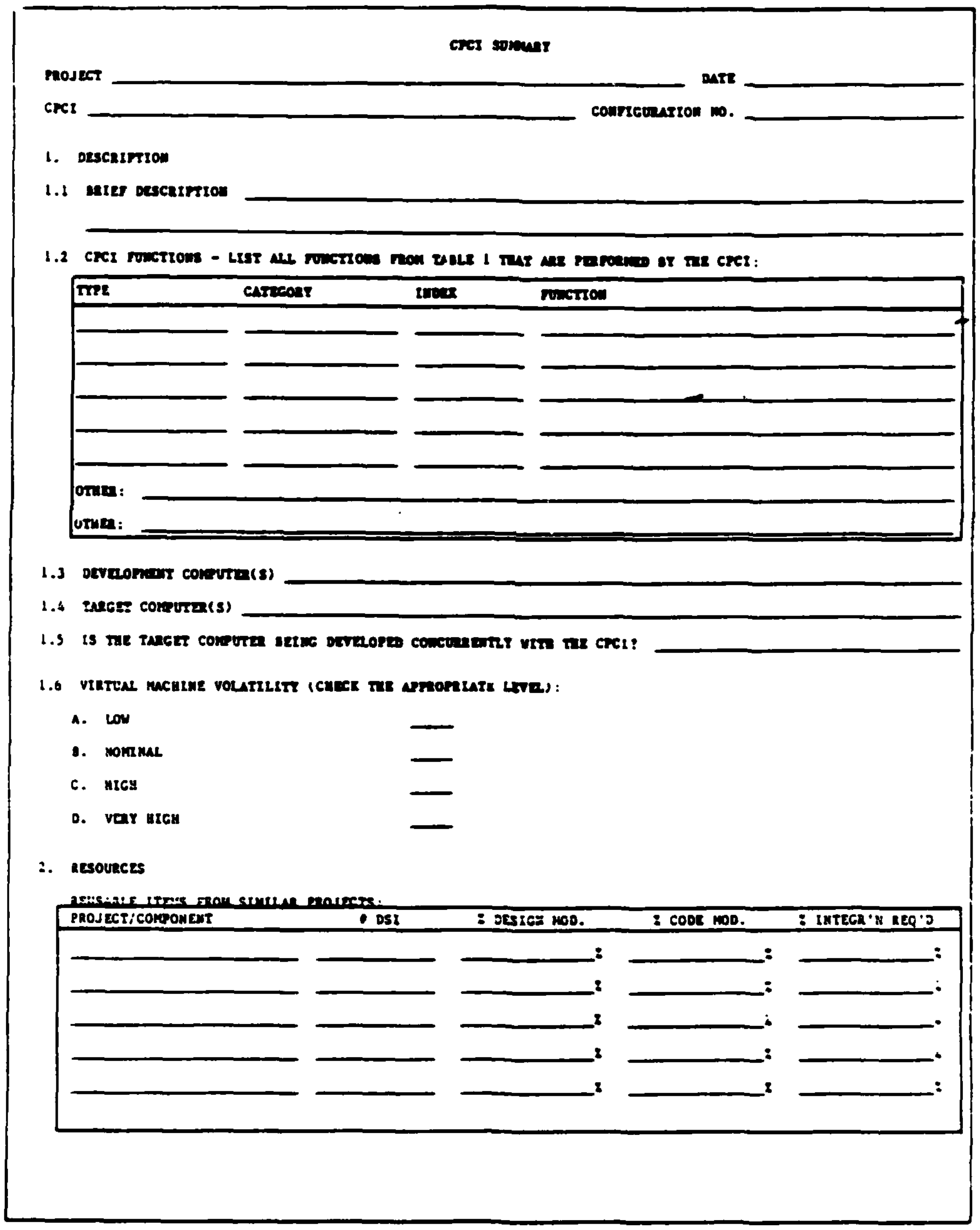

LEVEL 1
DEFENSE SYSTEM
LEVEL 2
PRIME MISSION EQUIPMENT
PRIME MISSION SOFTWARE
TRAINING
LEVEL 3
SOFTWARE SYSTEM ANALYSIS & DESIGN
SOFTWARE SYSTEM INTEGRATION & TEST
SOFTWARE SUBSYSTEM
SOFTWARE SUBSYSTEM
SUPPORT SOFTWARE
LEVEL 4
SUBSYSTEM ANALYSIS & DESIGN
SUBSYSTEM INTEGRATION & TEST
CPCI
CPCI
REQUIREMENTS DEFINITION (DEVELOPMENT SPECIFICATION)
DESIGN (PRODUCT SPECIFICATION)
CPC
· CPC DESIGN
· CPC CODE AND DEBUG
· CPC INTEGRATION AND CHECKOUT
DATA GENERATION/CONVERSION
INTEGRATION & INFORMAL TEST
QUALIFICATION TESTS
CPCI DOCUMENTS
SYSTEM TEST SUPPORT AND INITIAL MAINTENANCE

Figure 4-2. CPCI Work Breakdown Structure Elements [25]

and encompasses the following eleven areas:

- o Project Description
- o Resources
- o Total System Size
- o Difficulty
- o Techniques Employed
- o Formalisms Used
- o Automated Tools Used
- o Software Standards
- o Project Schedule
- o System-level Software-Related Documentation
- o Corporate Experience

The CPCI Summary form is shown in figure 4-3.[17] It is directly compatible with the input requirements of the COCOMO model and many instructions for completing the form are directly from the Boehm book. A key input to the form is a breakout of all the software functions performed by each CPCI.



CPCI SUMMARY

PROJECT ______________________ DATE __________

CPCI ______________________ CONFIGURATION NO. __________

1. DESCRIPTION

1.1 BRIEF DESCRIPTION ______________________

1.2 CPCI FUNCTIONS - LIST ALL FUNCTIONS FROM TABLE 1 THAT ARE PERFORMED BY THE CPCI:

TYPE	CATEGORY	INDEX	FUNCTION
OTHER:			
OTHER:			

1.3 DEVELOPMENT COMPUTER(S) ______________________

1.4 TARGET COMPUTER(S) ______________________

1.5 IS THE TARGET COMPUTER BEING DEVELOPED CONCURRENTLY WITH THE CPCI? __________

1.6 VIRTUAL MACHINE VOLATILITY (CHECK THE APPROPRIATE LEVEL):

A. LOW ____

B. NOMINAL ____

C. HIGH ____

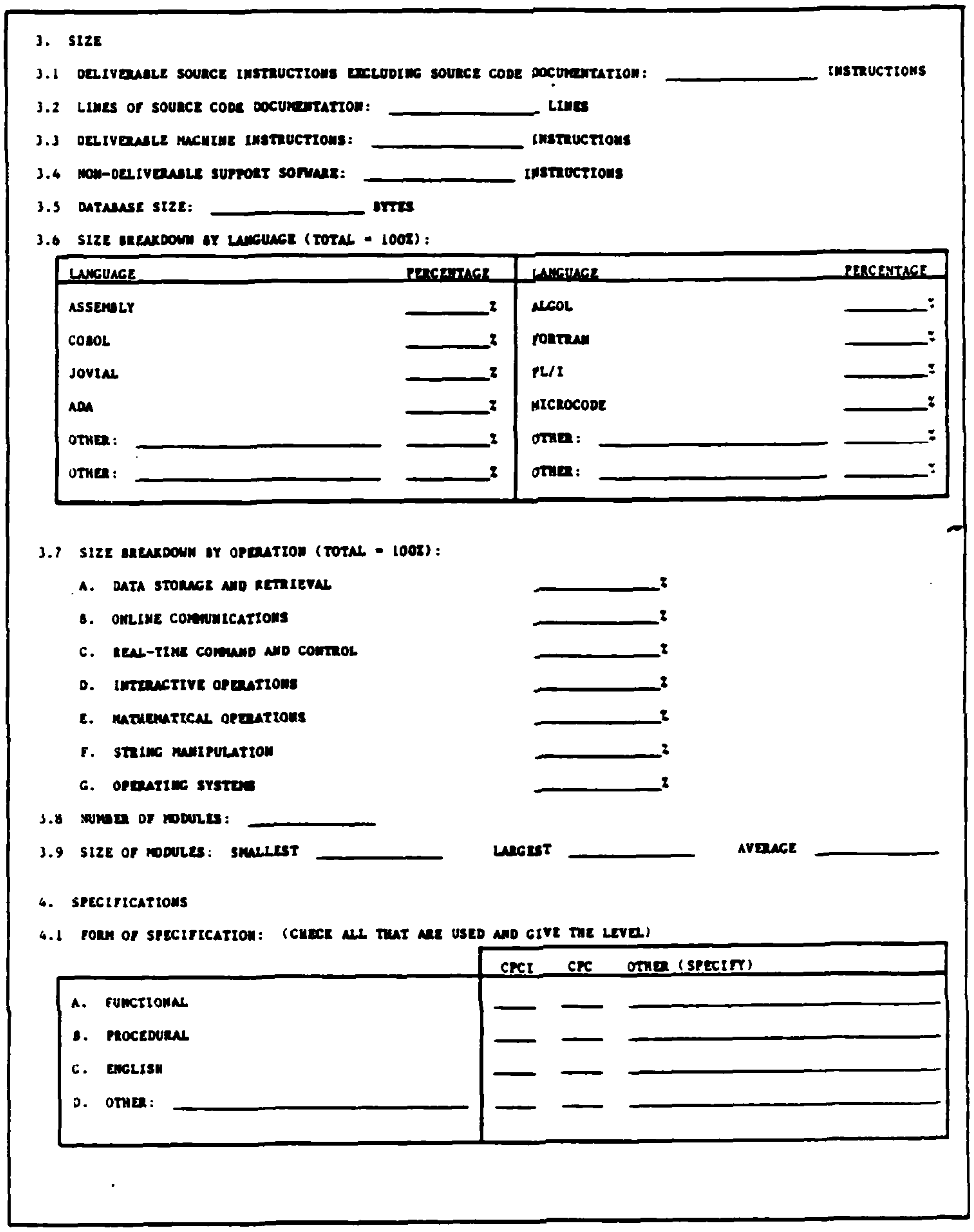
D. VERY HIGH ____

2. RESOURCES

REUSABLE ITEMS FROM SIMILAR PROJECTS:

PROJECT/COMPONENT	# DSI	% DESIGN MOD.	% CODE MOD.	% INTEGR'N REQ'D
		%	%	%
		%	%	%
		%	%	%
		%	%	%
		%	%	%

Figure 4-3. CPCI Summary Form[14]



3. SIZE

3.1 DELIVERABLE SOURCE INSTRUCTIONS EXCLUDING SOURCE CODE DOCUMENTATION: ____________ INSTRUCTIONS

3.2 LINES OF SOURCE CODE DOCUMENTATION: ____________ LINES

3.3 DELIVERABLE MACHINE INSTRUCTIONS: ____________ INSTRUCTIONS

3.4 NON-DELIVERABLE SUPPORT SOFWARE: ____________ INSTRUCTIONS

3.5 DATABASE SIZE: ____________ BYTES

3.6 SIZE BREAKDOWN BY LANGUAGE (TOTAL = 100%):

LANGUAGE	PERCENTAGE	LANGUAGE	PERCENTAGE
ASSEMBLY	______%	ALGOL	______%
COBOL	______%	FORTRAN	______%
JOVIAL	______%	PL/I	______%
ADA	______%	MICROCODE	______%
OTHER: ____________	______%	OTHER: ____________	______%
OTHER: ____________	______%	OTHER: ____________	______%

3.7 SIZE BREAKDOWN BY OPERATION (TOTAL = 100%):

A. DATA STORAGE AND RETRIEVAL	________%
B. ONLINE COMMUNICATIONS	________%
C. REAL-TIME COMMAND AND CONTROL	________%
D. INTERACTIVE OPERATIONS	________%
E. MATHEMATICAL OPERATIONS	________%
F. STRING MANIPULATION	________%
G. OPERATING SYSTEMS	________%

3.8 NUMBER OF MODULES: ____________

3.9 SIZE OF MODULES: SMALLEST ____________ LARGEST ____________ AVERAGE ____________

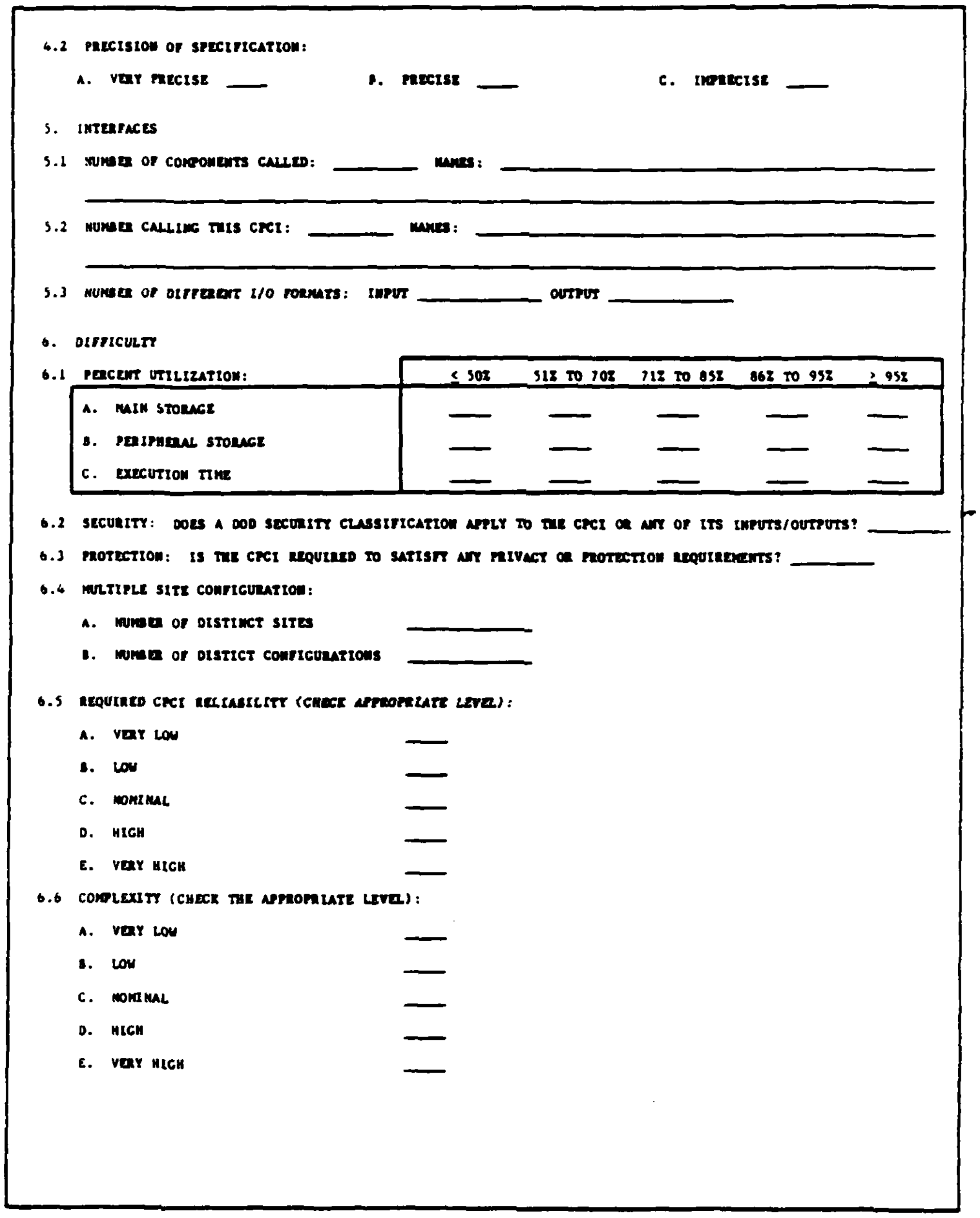
4. SPECIFICATIONS

4.1 FORM OF SPECIFICATION: (CHECK ALL THAT ARE USED AND GIVE THE LEVEL)

	CPCI	CPC	OTHER (SPECIFY)
A. FUNCTIONAL	___	___	____________
B. PROCEDURAL	___	___	____________
C. ENGLISH	___	___	____________
D. OTHER: ____________	___	___	____________

14

Figure 4-3. CPCI Summary Form (cont.)



6.2 PRECISION OF SPECIFICATION:

A. VERY PRECISE ____ B. PRECISE ____ C. IMPRECISE ____

5. INTERFACES

5.1 NUMBER OF COMPONENTS CALLED: ________ NAMES: ____

5.2 NUMBER CALLING THIS CPCI: ________ NAMES: ____

5.3 NUMBER OF DIFFERENT I/O FORMATS: INPUT ____________ OUTPUT ____________

6. DIFFICULTY

6.1 PERCENT UTILIZATION:

	$\leq$ 50%	51% TO 70%	71% TO 85%	86% TO 95%	$\geq$ 95%
A. MAIN STORAGE	____	____	____	____	____
B. PERIPHERAL STORAGE	____	____	____	____	____
C. EXECUTION TIME	____	____	____	____	____

6.2 SECURITY: DOES A DOD SECURITY CLASSIFICATION APPLY TO THE CPCI OR ANY OF ITS INPUTS/OUTPUTS? ________

6.3 PROTECTION: IS THE CPCI REQUIRED TO SATISFY ANY PRIVACY OR PROTECTION REQUIREMENTS? ________

6.4 MULTIPLE SITE CONFIGURATION:

A. NUMBER OF DISTINCT SITES ____________

B. NUMBER OF DISTICT CONFIGURATIONS ____________

6.5 REQUIRED CPCI RELIABILITY (CHECK APPROPRIATE LEVEL):

A. VERY LOW ____

B. LOW ____

C. NOMINAL ____

D. HIGH ____

E. VERY HIGH ____

6.6 COMPLEXITY (CHECK THE APPROPRIATE LEVEL):

A. VERY LOW ____

B. LOW ____

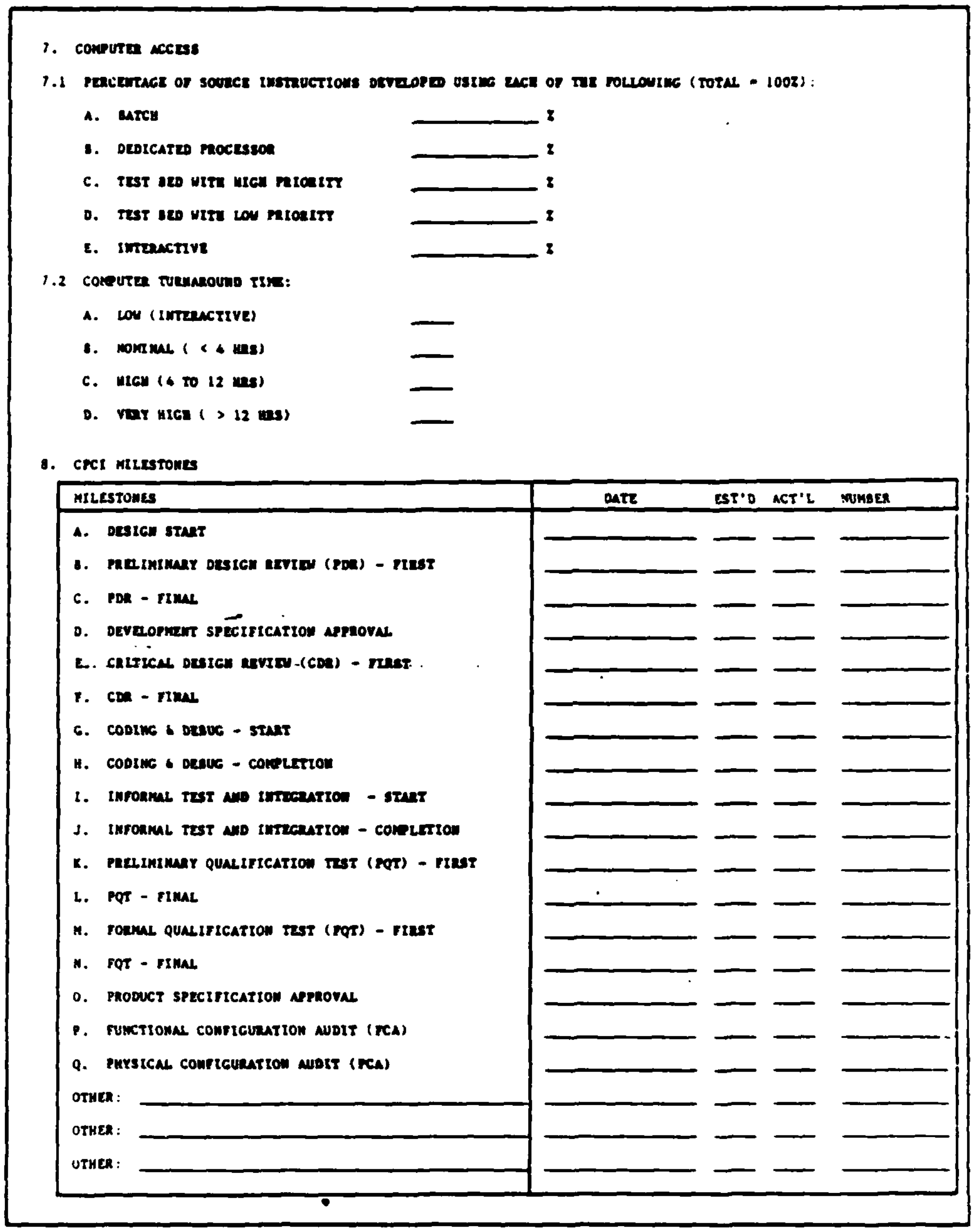
C. NOMINAL ____

D. HIGH ____

E. VERY HIGH ____

14

Figure 4-3. CPCI Summary Form (cont.)



7. COMPUTER ACCESS

7.1 PERCENTAGE OF SOURCE INSTRUCTIONS DEVELOPED USING EACH OF THE FOLLOWING (TOTAL = 100%):

A. BATCH	________	%
B. DEDICATED PROCESSOR	________	%
C. TEST BED WITH HIGH PRIORITY	________	%
D. TEST BED WITH LOW PRIORITY	________	%
E. INTERACTIVE	________	%

7.2 COMPUTER TURNAROUND TIME:

A. LOW (INTERACTIVE)	___
B. NOMINAL (< 4 HRS)	___
C. HIGH (4 TO 12 HRS)	___
D. VERY HIGH (> 12 HRS)	___

8. CPCI MILESTONES

MILESTONES	DATE	EST'D	ACT'L	NUMBER
A. DESIGN START				
B. PRELIMINARY DESIGN REVIEW (PDR) - FIRST				
C. PDR - FINAL				
D. DEVELOPMENT SPECIFICATION APPROVAL				
E. CRITICAL DESIGN REVIEW (CDR) - FIRST				
F. CDR - FINAL				
G. CODING & DEBUG - START				
H. CODING & DEBUG - COMPLETION				
I. INFORMAL TEST AND INTEGRATION - START				
J. INFORMAL TEST AND INTEGRATION - COMPLETION				
K. PRELIMINARY QUALIFICATION TEST (PQT) - FIRST				
L. PQT - FINAL				
M. FORMAL QUALIFICATION TEST (FQT) - FIRST				
N. FQT - FINAL				
O. PRODUCT SPECIFICATION APPROVAL				
P. FUNCTIONAL CONFIGURATION AUDIT (FCA)				
Q. PHYSICAL CONFIGURATION AUDIT (PCA)				
OTHER:				
OTHER:				
OTHER:				

14

Figure 4-3. CPCI Summary Form (cont.)

9. DOCUMENTATION

TITLE	DELIVERY DATE	# PAGES EST'D	ACT'L
A. CPCI DEVELOPMENT SPECIFICATION			
B. CPCI PRODUCT SPECIFICATION			
C. TEST PLAN			
D. TEST PROCEDURES			
E. TEST REPORT			
F. USER'S MANUAL			
G. OTHER:			
H. OTHER:			
I. OTHER:			
J. OTHER:			

10. PERSONNEL

10.1 AVERAGE EXPERIENCE OF PERSONNEL

	≤ 4 MOS	4 MOS TO 1 YR	1 TO 3 YRS	3 TO 6 YRS	≥ 6 YEARS
A. APPLICATION AREA					
B. TECHNIQUES TO BE USED					
C. LANGUAGES TO BE USED					
D. VIRTUAL MACHINE					
E. SUPPORT SOFTWARE/TOOLS					

10.2 AVERAGE QUALITY OF THE CPCI DEVELOPMENT PERSONNEL (PERCENTILES):

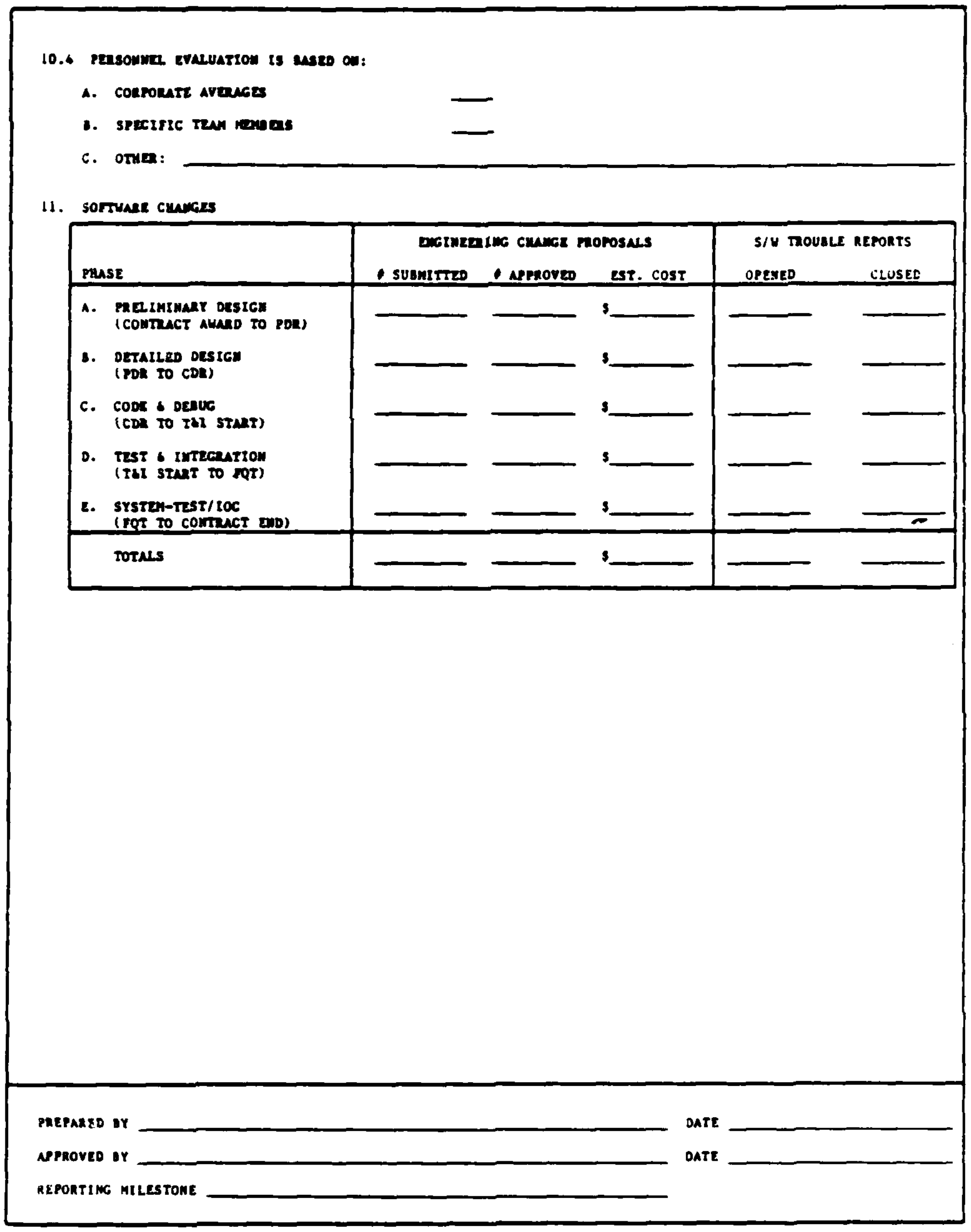
	≤ 15%	16 - 35%	36 - 55%	56 - 75%	76 - 90%	≥ 90%
A. ANALYSTS/DESIGNERS						
B. PROGRAMMERS						
C. TESTERS						
D. OVERALL						

10.3 EXPERIENCE WITH MODERN PROGRAMMING PRACTICES:

A. VERY LOW	
B. LOW	
C. NOMINAL	
D. HIGH	
E. VERY HIGH	

14

Figure 4-3. CPCI Summary Form (cont.)



10.4 PERSONNEL EVALUATION IS BASED ON:

A. CORPORATE AVERAGES ____

B. SPECIFIC TEAM MEMBERS ____

C. OTHER: ____________________

11. SOFTWARE CHANGES

	ENGINEERING CHANGE PROPOSALS			S/W TROUBLE REPORTS	
PHASE	# SUBMITTED	# APPROVED	EST. COST	OPENED	CLOSED
A. PRELIMINARY DESIGN (CONTRACT AWARD TO PDR)	____	____	$____	____	____
B. DETAILED DESIGN (PDR TO CDR)	____	____	$____	____	____
C. CODE & DEBUG (CDR TO T&I START)	____	____	$____	____	____
D. TEST & INTEGRATION (T&I START TO FQT)	____	____	$____	____	____
E. SYSTEM-TEST/IOC (FQT TO CONTRACT END)	____	____	$____	____	____
TOTALS	____	____	$____	____	____

PREPARED BY ____________________ DATE ________

APPROVED BY ____________________ DATE ________

REPORTING MILESTONE ____________________

14

Figure 4-3. CPCI Summary Form (cont.)

A table of generic operational and support functions is provided for uniformity of data classification. That table is shown in Table 4-1.[18]

Table 4-1. Software Functions

Type	Category	Index	Function
Operational	Displays	1.1	Avionics
		1.2	Command, Control, & Communications
		1.3	Other
	Avionics	2.1	Mission Planning
		2.2	Navigation
		2.3	Aircraft Steering & Flight Control
		2.4	Sighting, Designation & Location Determination
		2.5	Weapon Delivery
		2.6	Electronic Countermeasures
		2.7	Other
	Command, Control, & Communication	3.1	Network Monitoring
		3.2	Network Control & Switching
		3.3	Sensor Control
		3.4	Signal Processing
		3.5	Message Processing
		3.6	Message Distribution
		3.7	Message Logging & Retrieval
		3.8	Data Reduction
		3.9	Other
	Executive	4.1	Computer Resource Management
		4.2	Computer Operator Interface
		4.3	Other Terminal Operator Interface
		4.4	Special Device Interface
		4.5	Other Input or Output
		4.6	Error Handling/Reconfiguration/Recovery
		4.7	Multicomputer Configuration Control
		4.8	Performance Monitoring & Data Collection
		4.9	Other
	Data Base	5.1	On-Line Data Base Retrieval & Output
		5.2	On-Line Data Base Initialization & Updating
		5.3	Other
	Training	6.1	Control of Exercise Sequencing
		6.2	Operator Performance Data Collection
		6.3	Other
	On-Line Equipment Diagnostic	7.1	System Readiness Test
		7.2	Computer Diagnostic
		7.3	Memory Diagnostic
		7.4	Display Diagnostic
		7.5	Switch/Indicator Panel Diagnostic
		7.6	I/O Diagnostic
		7.7	Mode Diagnostic
		7.8	Other

Table 4-1. Software Functions (cont.)

Type	Category	Index	Function
Support	Operating System	8.1	Computer Resource Management
		8.2	Computer Operator Interface
		8.3	Terminal Operator Interface
		8.4	Input or Output
		8.5	Error Handling/Reconfiguration/Recovery
		8.6	Performance Monitoring & Data Collection
		8.7	Other
	Equipment Maintenance	9.1	Off-Line Computer Diagnostics
		9.2	Other
	Software Development	10.1	Higher-Order Language Compiler
		10.2	Assembler
		10.3	Debugger
		10.4	Loader or Editor
		10.5	Other
	Off-Line Data Base Management	11.1	Data Base Definition
		11.2	Data Base Initialization or Updating
		11.3	Data Base Retrieval & Output Formatting
		11.4	Data Base Restructuring
		11.5	Off-Line Data Base
		11.6	Other
	Design	12.1	Data Base Design
		12.2	Data Base Processor Design
		12.3	Performance Simulation
		12.4	Data Reduction
		12.5	Data Analysis
		12.6	Other
	Test Software	13.1	Test Case Generation
		13.2	Test Case Data Recording
		13.3	Test Data Reduction
		13.4	Test Analysis
		13.5	Other
	Utilities	14.1	Media Conversions
		14.2	Format Translation
		14.3	Sort/Merge
		14.4	Program Library Maintenance
		14.5	Other
	Off-Line Training	15.1	Data Reduction
		15.2	Training Analysis
		15.3	Scenario Preparation
		15.4	Other
	Project Management	16.1	Project Event Status Accounting
		16.2	Schedule Maintenance/Projection
		16.3	Financial Accounting
		16.4	Software Cost Reporting
		16.5	Hardware Cost Reporting
		16.6	Software Cost Prediction
		16.7	Hardware Cost Prediction
		16.8	Other
	Hardware Subsystem Simulations	17.1	Interfacing Hardware Simulations
		17.2	Environmental Simulations
		17.3	Operator Action Simulations
		17.4	Other

5. CONCEPT DEVELOPMENT

From each area researched selected features were integrated into a concept for an interactive software cost estimating model compatible with the amount of information available at the conceptual phase of the life cycle. Figure 5-1 illustrates the features selected.

5.1 Selected Features

The heart of the concept is the COCOMO software cost estimating equations. These equations are input by analogous judgments made from reviews of stored libraries of baseline C3I system software. The database structure used is a combination of the data structures used by the SARE breakdown developed by MITRE and the formats of the DACS center. The WICOMO "help" screen approach is the bases for parameter inputting, and the Navy HARDMAN concept of default libraries will be the basis for sizing analogies and COCOMO calibration. Finally, there will be compatibility with the NASA/SEL dataset generic

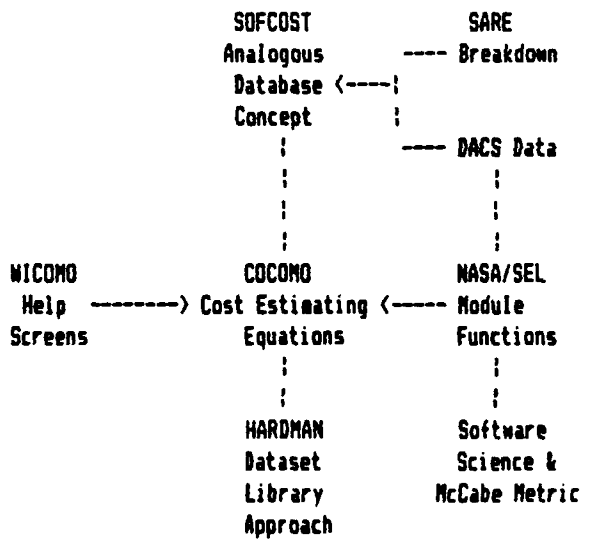

SOFCOST Analogous Database Concept
SARE Breakdown
DACS Data
WICOMO Help Screens
COCOMO Cost Estimating Equations
NASA/SEL Module Functions
HARDMAN Dataset Library Approach
Software Science & McCabe Metric

Figure 5-1. Concepts Selected

module structure and the metrics of software science and cyclomatics.

5.2 Dataset Formats

Software design trade-off data set formats were formulated along the basis of those used for hardware in the Hardware models. These are the sets of data that are compatible with a cost estimate made using the COCOMO model. Three levels of software breakdown were formulated: modules, CPCs, and CPCIs. A module is defined to be compatible with the SARE as a discrete part of a computer program configuration item (CPCI) with an identifiable function and which can be individually compiled or assembled.

Modules are grouped into six generic classes to serve as building blocks of software code from which CPCs and CPCIs can be designed. These six classes are compatible with those identified by the NASA Software Engineering Laboratory:

1) System Related
2) Input/Output Processing
3) Algorithmic
4) Logic Control
5) Data/COMMON Block
6) Other

A generic "system related" module is defined as a logical block of code used for operating systems, executive programs, and task management programs. "Input/Output Processing" modules are logical blocks of code related to activities external to the computation system. "Algorithmic" modules are logical blocks of code used for calculations of formulas, equations, trigonometry, and data manipulation in general. "Logic Control" modules are logical blocks of code used for making decisions based on previously stored/manipulated data. "Data/COMMON Block" modules are blocks of instructions related to those constants, either volatile or nonvolatile, which are set by the programmer. Figure 5-2 shows the structure of the Module level dataset format.

A computer program component (CPC) is a grouping of software modules into logically distinct parts of a CPCI distinguished for convenience in design and specification. Figure 5-3 shows the structure of the CPC level dataset format.

Figure 5-4 shows the structure of the CPCI level report. A CPCI is an aggregation of software computer program components which satisfies an end use function and has been designated for configuration management.

Figures 5-2, 5-3, and 5-4 are the input and output information that will be programmed to appear on the display software cost estimating workstation terminal and be printed out in hardcopy. The acronyms in bold print define the input data "help" screens that will be programmed and called by typing that acronym. Any input will be changable and the resulting changes will automatically be made in the associated software life cycle cost estimate.

The output of the estimate is a software life cycle cost and its associated estimating parameters. Software life cycle costs are defined as consisting of Development, Implementation, and Maintenance costs. The elements of development cost are

Plans and Requirement costs, Product Design costs, Programming costs, Integration and Test cost, and the cost of computer time used in program development and test. Implementation costs are the costs of computer program installation and operator training. Maintenance costs are the costs of computer program update and repair (debugging). The parameter summary gives the basis for the life cycle cost estimate resulting from using the inputs in the COCOMO model equations. KEDSI is thousands of "equivalent" delivered source instructions. Development MM are the total person months required for development. Annual Maintenance MM are the total person months required annually for computer program maintenance. The Nominal Development and Implementation times are the calendar months required for those functions. At the CPC level the modules in the CPC are tabulated along with their KEDSI and development and annual maintenance person months. Similarly at the CPCI level the CPC within a CPCI are tabulated.

```
Software Module
**********************************************************************

Function:


COST SUMMARY (      YR $000)         PARAMETER SUMMARY
********************************     **********************************

LIFE CYCLE COST             ______    KEDSI                          ______
                                      Development MM                 ______
DEVELOPMENT COST            ______    Computer Time                  ______
                                      Annual Maintenance MM          ______
Plans and Requirement______           Nominal Development Time       ______M
Product Design       ______           Nominal Implementation Time    ______M
Programming          ______           Length of Operating Life       ______YR
Integration and Test ______
Computer Time        ______

IMPLEMENTATION COSTS        ______    MAINTENANCE COSTS              ______

Installation         ______           Update                     ______
Training             ______           Repair                     ______

INPUT DATA
**********
1  DEVC   Development Computer Type (MAXI,MIDI,MINI,MICR). . .    ______
2  MODE   Software Development Mode (ORGN,SEMI,EMED) . . . . .    ______
3  KDSI   Thousands of Delivered Source Instructions(decimal).    ______
4  ADPT   Percent KDSI Adapted (integer) . . . . . . . . . . .    ______
5  CPI    Conversion Planning Increment (integer). . . . . . .    ______
6  DM     Percent Design Modified (integer). . . . . . . . . .    ______
7  CM     Percent Code Modified (integer). . . . . . . . . . .    ______
8  IM     Percent Integration Required for Mod. (integer). . .    ______
9  COMP   Computer hrs/mm of Development (decimal) . . . . . .    ______
10 CPLX   Module Complexity (decimal). . . . . . . . . . . . .    ______
11 RELY   Required Module Reliability (decimal). . . . . . . .    ______
12 PCAP   Programmer Capability (decimal). . . . . . . . . . .    ______
13 VEXP   Virtual Machine Experience (decimal) . . . . . . . .    ______
14 LEXP   Programming Language Experience (decimal). . . . . .    ______
15 AEXP   Applications Experience (decimal). . . . . . . . . .    ______
16 INST   Installation Complexity (decimal). . . . . . . . . .    ______
17 TRAIN  Training Complexity (decimal). . . . . . . . . . . .    ______
18 ACT    Annual Change Traffic (decimal). . . . . . . . . . .    ______
```

Figure 5-2 Computer Program Module Dataset

Software CPC
****

Function:

COST SUMMARY (YR $000)
****

LIFE CYCLE COST	______
DEVELOPMENT COST	______
Plans and Requirement	______
Product Design	______
Programming	______
Integration and Test	______
Computer Time	______
IMPLEMENTATION COSTS	______
Installation	______
Training	______
MAINTENANCE COSTS	______
Update	______
Repair	______

PARAMETER SUMMARY
****

MODULE	QTY	KEDSI	MM DEV	MM AM
System	___	______	______	______
I/O	___	______	______	______
Algor.	___	______	______	______
Logic	___	______	______	______
D/B	___	______	______	______
Other	___	______	______	______
TOTAL	___	______	______	______

Nominal Development Time	______M
Nominal Implementation Time	______M
Length of Operating Life	______YR

INPUT DATA
****

1	DEVC	Development Computer Type (MAXI,MIDI,MINI,MICR). . .	______
2	MODE	Software Development Mode (ORGN,SEMI,EMED)	______
3	KDSI	Thousands of Delivered Source Instructions(decimal).	______
4	ADPT	Percent KDSI Adapted (integer)	______
5	CPI	Conversion Planning Increment (integer).	______
6	DM	Percent Design Modified (integer).	______
7	CM	Percent Code Modified (integer).	______
8	IM	Percent Integration Required for Mod. (integer). . .	______
9	COMP	Computer hrs/mm of Development (decimal)	______
10	CPLX	Module Complexity (decimal).	______
11	RELY	Required Module Reliability (decimal).	______
12	TIME	Execution Time Constraint (decimal).	______
13	STOR	Main Storage Constraint (decimal).	______
14	DATA	Data Base Size Factor (decimal).	______
15	ACT	Annual Change Traffic (decimal).	______

Figure 5-3 Computer Program Component Dataset

```
Software CPCI
*********************************************************************

Function:

COST SUMMARY (      YR $000)          PARAMETER SUMMARY
********************************     **********************************
                                     CPC    QTY   KEDSI    MM        MM
                                                            DEV        AM
LIFE CYCLE COST             ______   ___    ___   ______   ______    ______
                                     ___    ___   ______   ______    ______
DEVELOPMENT COST            ______   ___    ___   ______   ______    ______
                                     ___    ___   ______   ______    ______
Plans and Requirement______          ___    ___   ______   ______    ______
Product Design        ______         ___    ___   ______   ______    ______
Programming           ______
Integration and Test  ______         TOTAL  ___   ______   ______    ______
Computer Time         ______
                                     Nominal Development Time        ______M
IMPLEMENTATION COSTS        ______   Nominal Implementation Time     ______M
                                     Length of Operating Life        ______YR
Installation         ______
Training             ______

MAINTENANCE COSTS*          ______

Update               ______
Repair               ______
```

INPUT DATA
****

1	DEVC	Development Computer Type (MAXI,MIDI,MINI,MICR). . .	______
2	MODE	Software Development Mode (ORGN,SEMI,EMED)	______
3	KDSI	Thousands of Delivered Source Instructions(decimal).	______
4	ADPT	Percent KDSI Adapted (integer)	______
5	CPI	Conversion Planning Increment (integer).	______
6	DM	Percent Design Modified (integer).	______
7	CM	Percent Code Modified (integer).	______
8	IM	Percent Integration Required for Mod. (integer). . .	______
9	COMP	Computer hrs/mm of Development (decimal)	______
10	CPLX	Module Complexity (decimal).	______
11	RELY	Required Module Reliability (decimal).	______
12	VIRT	Virtual Machine Volatility (decimal)	______
13	TURN	Computer Turnaround Time (decimal)	______
14	ACAP	Analyst Capability (decimal)	______
15	MODP	Use of Modern Programmming Practices (decimal) . . .	______
16	TOOL	Use of Software Tools (decimal).	______
17	ACT	Annual Change Traffic (decimal).	______
18	YEAR	Dollars (then, now).	______

Figure 5-4 Computer Program Configuration Item Dataset

5.3 Dataset Inputs

The inputs at each level of the software hierarchy contain both common and unique data. The following common inputs are required regardless of the software structure level being estimated:

DEVC	-- the expected development computer (maxi, midi, mini, micro)
MODE	-- the expected software development mode as defined by Boehm (organic, semidetached, and embedded)
KDSI	-- the estimated number of thousands of delivered source instructions.
ADPI	-- the estimated percent of KDSI that could be adapted from existing programs.
CPI	-- the estimated planning increment of instructions needed to do the conversion analysis and planning.
DM	-- the estimated percent of existing programs that would be redesigned to perform the required functions and/or missions.
CM	-- the estimated percent of existing code required to be modified.
IM	-- the estimated percent of normal integration required for adapted software integration.
COMP	-- the estimated computer run time hours required to support a person-month of software development activity based on a type of computer and software product.
CPLX	-- the estimated relative effort multiplier based on complexity of the software program to be developed for the number of delivered source instructions.
RELY	-- the estimated relative effort of software development required for software reliability for a given number of delivered source instructions.
ACT	-- the estimated annual percent of effort required for software program source instruction change through additions or modifications.

At the module, but not the CPC and CPCI levels, the following information is input:

PCAP -- the estimated relative software production based on programmer capability.

VEXP -- the estimated relative software production based on programmer virtual machine experience.

LEXP -- the estimated relative software production based on the level of programming language experience of the project team developing the software module.

AEXP -- the estimated relative software production based on programmer experience with the software application.

INST -- the estimated percent of development effort required for software program installation and checkout.

TRAIN - the estimated percent of development effort required for software operator and maintenance support training.

At the CPC but not the CPCI or module levels, the following unique information is input:

TIME -- the estimated added effort required for a given number of instructions based on expected available execution time.

STOR -- the estimated added effort required for a given number of instructions based on program expected main storage usage.

DATA -- the estimated relative effort for the development of the size of the data base required.

At the CPCI but not the CPC or module levels, the following unique information is input:

VIRT -- the estimated virtual machine volatility impact on the effort required to develop a given number of delivered source instructions.

TURN -- the estimated computer turn-around time for program decks effect on the effort required to develop a given number of delivered source instructions.

ACAP -- the estimated impact of the software analysts capability on the effort required to develop a given number of delivered source instructions.

MODP -- the estimated impact of the amount of modern programming methods applied to the development on the effort required to develop a given number of delivered source instructions.

TOOL -- the estimated impact of the presupposed software tools that will be used on the effort required to develop a given number of delivered source instructions.

The programs that generate the datasets reports will be capable of running independently or additively, i.e., a run can be made at the CPCI level by inputting the CPCI level model, at the CPC level by inputting the CPC level model, or at the module level by inputting the module level model; or a run could be made of CPCs built from groups of modules, and CPCIs from groups of CPCs. At each level of the software breakdown Help screens have been developed to aid inputting, and at each level default data will be developed for all inputs. Default data will be contained in libraries of modules, CPCs, and CPCI life cycle cost data sets.

5.4 Cost Estimating Equations

The equations to be used to calculate estimates of life cycle cost are the equations of the COCOMO model with modifications to be compatible with the three-tiered software structure and an interactive computer program:

THOUSANDS EQUIVALENT DELIVERED SOURCE INSTRUCTIONS (KEDSI)

$$\text{MODULE KEDSI} = [(\text{ADPI}/100)(\text{KDSI})][1.0 + (0.40(\text{DM}) + 0.30(\text{CM}) + 0.30(\text{IM}) + \text{CPI})/100]$$

$$\text{CPC KEDSI} = \text{KEDSI}_{\text{MODULES}}$$

$$\text{CPCI KEDSI} = \text{KEDSI}_{\text{CPC}}$$

PERSON MONTHS (MM)

$$\text{ORGANIC MM}_{\text{NOM}} = 3.2(\text{KEDSI})^{1.05}$$

$$\text{SEMIDETACHED MM}_{\text{NOM}} = 3.0(\text{KEDSI}^{1.12}$$

$$\text{EMBEDDED MM}_{\text{NOM}} = 2.8(\text{KEDSI})^{1.20}$$

$$\text{MM}_{\text{DEV}} = [\text{MM}_{\text{NOM}}][\text{EAF}]$$

DEVELOPMENT EFFORT ADJUSTMENT FACTOR (EAF)

$$\text{MODULE EAF} = [(\text{PCAP})(\text{VEXP})(\text{LEXP})(\text{CPLX})(\text{RELY})(\text{AEXP})]$$

$$\text{CPC EAF} = [(\text{TIME})(\text{STOR})(\text{DATA})]$$

$$\text{CPCI EAF} = [(\text{VIRT})(\text{TURN})(\text{ACAP})(\text{MODP})(\text{TOOL})]$$

PHASE DISTRIBUTION OF DEVELOPMENT EFFORT (FRAC_{P})

$$\text{MM}_{\text{NOM,P}} = [(\text{KEDSI})((\text{FRAC}_{\text{P}})]/(\text{KEDSI})/(\text{MM}_{\text{NOM}})][\text{EAF}]$$

$$\text{COMPUTER TIME} = (\text{MM}_{\text{DEV}})(\text{CHR}/\text{MM}_{\text{DEV}})$$

IMPLEMENTATION

$$\text{INSTALLATION} = (\text{INST})(\text{MM}_{\text{DEV}})$$

$$\text{TRAINING} = (\text{TRAIN})(\text{MM}_{\text{DEV}})$$

ANNUAL MAINTENANCE PERSON-MONTH (MM_{AM})

$$\text{MM}_{\text{AM}} = (\text{ACT})(\text{MM}_{\text{NOM}})(\text{EAF}_{\text{M}})$$

MAINTENANCE EFFORT ADJUSTMENT FACTOR (EAF_{M})

$$\text{EAF}_{\text{M}} = [(\text{PCAP}_{\text{M}})(\text{VEXP}_{\text{M}})(\text{LEXP}_{\text{M}})(\text{CPLX}_{\text{M}})(\text{RELY}_{\text{M}})(\text{AEXP}_{\text{M}})]$$

MAINTENANCE ACTIVITY APPORTIONMENTS

$$\text{REPAIRS} = 45.3/100(\text{MM}_{\text{AM}})$$

$$\text{UPDATES} = 54.7/100(\text{MM}_{\text{AM}})$$

NOMINAL DEVELOPMENT TIME (TD)

$$\text{ORGANIC TD} = 2.5(\text{MM}_{\text{DEV}})^{0.38}$$

$$\text{SEMIDETACHED TD} = 2.5(\text{MM}_{\text{DEV}})^{0.35}$$

$$\text{EMBEDDED TD} = 2.5(\text{MM}_{\text{DEV}})^{0.32}$$

NOMINAL IMPLEMENTATION TIME (TI)

$$\text{ORGANIC TI} = 3.2(MM_{DEV})^{1.05}$$

$$\text{SEMIDETACHED TI} = 3.0(MM_{DEV})^{1.12}$$

$$\text{EMBEDDED TI} = 2.8(MM_{DEV})^{1.20}$$

5.5 Help Screens

The following HELP screens will be developed for interactive inputting:

1) MODE

This screen will provide help in determining the expected software development mode. It will appear on the screen as follows:

MODE = Software Development Mode.

MODE	CHARACTERISTICS	EXAMPLES	INPUT
ORGANIC	Less than 50KDSI	Scientific Models	ORGN
	Minimal Innovation	Business Models	
	Loosely Structured	Familiar OS/Compilers	
SEMIDETACHED	Less than 300KDSI	Training Simulators	SEMI
	Moderate Innovation	Transaction Processors	
	Moderately Structured	New OS/DBMS	
EMBEDDED	All sizes	Complex Simulators	EMBD
	Innovative	Real Time Processors	
	Tightly Structured	Command and Control	

2) CPI

This screen will provide help in determining the Conversion Planning Increment to cover the added costs of feasibility analysis and planning of existing software for a new application not included in adaptation estimates. It will appear as follows:

CPI = Conversion Planning Increment

LEVEL OF CONVERSION ANALYSIS AND PLANNING	INPUT
None	0
Simple schedule, acceptance plan	1
Detailed schedule, test, acceptance plans	2
Basic analysis of inventory of code, data	3
Detailed inventory plus basic documentation	4
Detailed inventory plus detailed documentation	5

3) DM

This screen will provide help in determining the percent of the adapted software's design which will be modified in order to adapt it to the new objectives and environment. It will appear as follows:

DM = Percent Design Modified

LEVEL OF ADAPTED DESIGN MODIFICATION	INPUT
None	0
Change to accommodate different doctrine	5
Change to accommodate overlay structure	10
Change to overlay structure, analogs, logic	15
Different formats, protocols, equipment	50

4) CM

This screen will provide help in determining the percentage of adapted software's code which will be modified in order to adapt it to new objectives and environment. It will appear as follows:

CM = Percent Code Modified

LEVEL OF ADAPTED CODE MODIFICATION	INPUT
None	0
Slight compiler differences & operating system interfaces	15
Change of word size	30
Different formats, protocol, equipment	60

5) IM

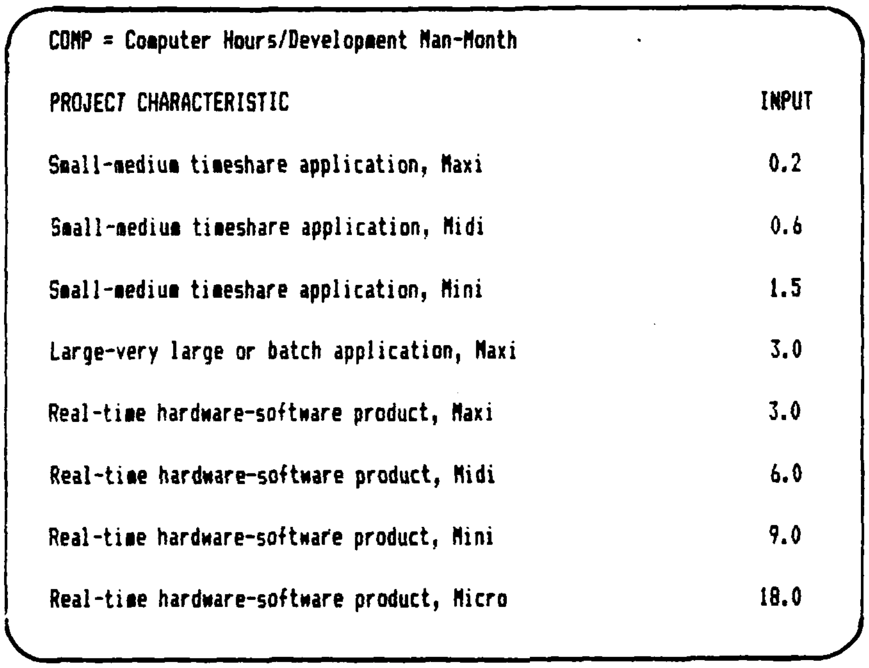
This screen will provide help in determining the percentage of effort required to integrate adapted software into an overall product, as compared to the normal amount of integration effort for software of comparable size. It will appear as follows:

IM = Percent Integration for Modification

LEVEL OF ADAPTED CODE INTEGRATION	INPUT
None	0
Minor Code Changes	5
Overlay/Word Size Changes	10
Test Data Integration	25
Different Formats and Displays	80

6) COMP

This screen will provide help in determining an estimate of computer run time hours required to support a man-month of software development activity. It will appear as follows:

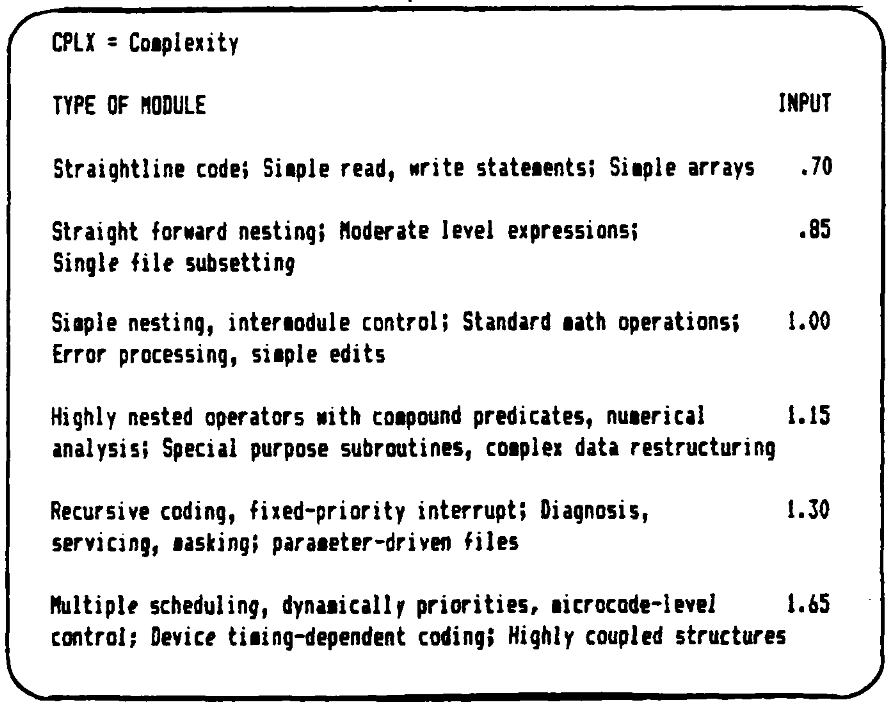


COMP = Computer Hours/Development Man-Month

PROJECT CHARACTERISTIC	INPUT
Small-medium timeshare application, Maxi	0.2
Small-medium timeshare application, Midi	0.6
Small-medium timeshare application, Mini	1.5
Large-very large or batch application, Maxi	3.0
Real-time hardware-software product, Maxi	3.0
Real-time hardware-software product, Midi	6.0
Real-time hardware-software product, Mini	9.0
Real-time hardware-software product, Micro	18.0

7) CPLX

This screen will provide help in determining an effort multiplier based on complexity of the software program to be developed. It will appear as follows:



CPLX = Complexity

TYPE OF MODULE	INPUT
Straightline code; Simple read, write statements; Simple arrays	.70
Straight forward nesting; Moderate level expressions; Single file subsetting	.85
Simple nesting, intermodule control; Standard math operations; Error processing, simple edits	1.00
Highly nested operators with compound predicates, numerical analysis; Special purpose subroutines, complex data restructuring	1.15
Recursive coding, fixed-priority interrupt; Diagnosis, servicing, masking; parameter-driven files	1.30
Multiple scheduling, dynamically priorities, microcode-level control; Device timing-dependent coding; Highly coupled structures	1.65

8) RELY

This screen will provide help in determining the relative effort of software development required for software reliability for a given number of delivered source instructions. It will appear as follows:

RELY = Software Reliability

EFFECT OF SOFTWARE FAILURE	EXAMPLE	INPUT
Inconvenience of fix	Demonstration prototype; Feasibility-phase simulation	0.75
Easily-recoverable loss to users.	Planning model or forecasting model.	0.88
Moderate loss; Recover with penalty	Management information or inventory control systems	1.00
Major loss or inconvenience	Accounting systems & power distribution systems	1.15
Loss of human life.	Military command and control systems	1.40

9) PCAP

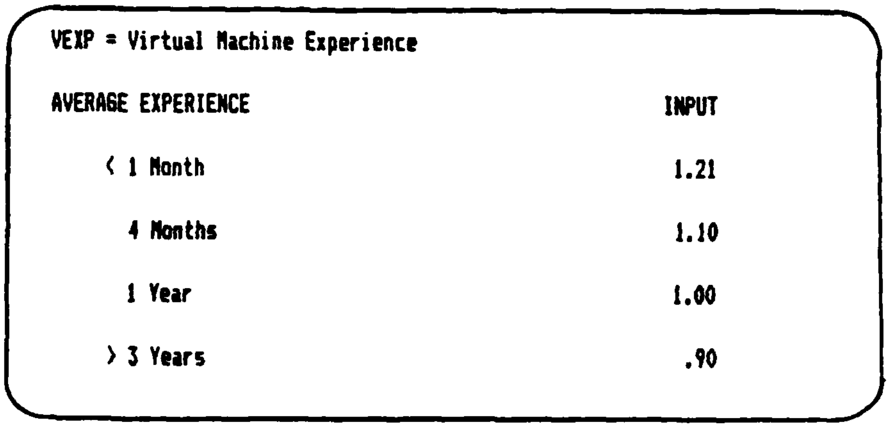
This screen will provide help in determining the estimated relative software production based on programmer capability. It will appear as follows:

PCAP = Programmer Capability (Team)

RELATIVE EFFICIENCY AND THOROUGHNESS		INPUT
Very Low	15%	1.42
Low	35%	1.17
Nominal	55%	1.00
High	75%	.86
Very High	90%	.70

10) VEXP

This screen will provide help in determining the estimated relative software production based on project team's virtual machine experience. It will appear as follows:

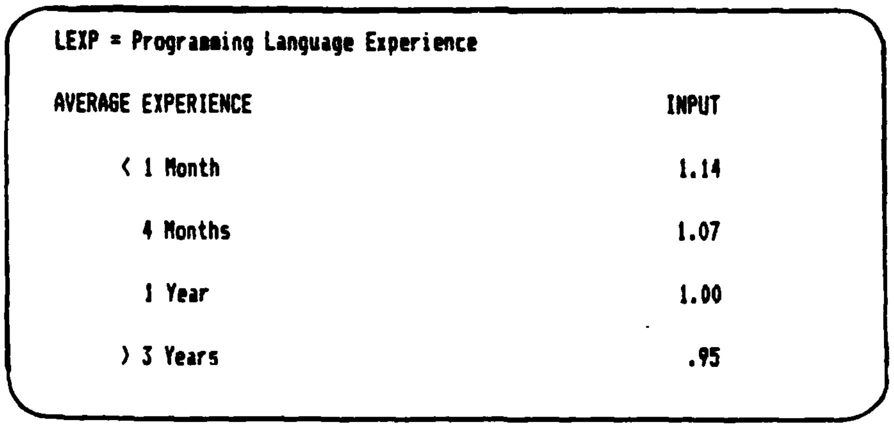


VEXP = Virtual Machine Experience

AVERAGE EXPERIENCE	INPUT
< 1 Month	1.21
4 Months	1.10
1 Year	1.00
> 3 Years	.90

11) LEXP

This screen will provide help in determining the estimated relative software production based on the level of programming language experience of the project team developing the software module. It will appear as follows:



LEXP = Programming Language Experience

AVERAGE EXPERIENCE	INPUT
< 1 Month	1.14
4 Months	1.07
1 Year	1.00
> 3 Years	.95

12) AEXP

This screen will provide help in determining the level of applications experience of the project team developing the proposed software. It will appear as follows:

AEXP = Applications experience

AVERAGE EXPERIENCE	INPUT
< 4 Months	1.29
1 Year	1.13
3 Years	1.00
6 Years	.91
> 12 Years	.82

13) INST

This screen will provide help in determining the estimated percent of development effort required for software program installation. It will appear on the screen as follows:

INST = Installation Effort

TYPE SOFTWARE	INPUT
Application program on existing general purpose computer	.2
Application program on different general-purpose computer	.8
Process control program on new computer	3
Human-machine system	13

14) TRAIN

This screen will provide help in determining the estimated percent of develpment effort required for newly installed software programs. It will appear on the screen as follows:

TRAIN = Training Effort

TYPE SOFTWARE	INPUT
Application program on existing general-purpose computer	1
Application program on different general-purpose computer	3
Process control program on new computer	4
Human-machine system	6

15) ACT

This screen will provide help in determining the fraction of source instructions which undergo change during a typical year either through additions or modifications. It will appear as follows:

ACT = Annual Change Traffic

TYPE OF SOFTWARE	INPUT
Non real-time input/output	.01
Mathematical and logical operations	.05
File, data base manipulation, real-time control	.08
Complex process control system	.20
Real-time command and control	.40

16) TIME

This screen will provide help in determining the added effort required for a given number of instructions based on execution time required. It will appear as follows:

TIME = Execution Time Required

REQUIRED TIME	INPUT
< 50%	1.00
70%	1.11
85%	1.30
95%	1.66

17) STOR

This screen will provide help in determining the estimated added effort required for a given number of instructions based on main storage usage. It will appear as follows:

STOR = Main Storage Required

REQUIRED MEMORY	INPUT
Nominal < 50%	1.00
High 70%	1.06
Very High 85%	1.21
Extra High 95%	1.56

18) DATA

This screen will help in determining the increased effort for development of the data base required to support the proposed program. It will appear as follows:

DATA = Data Base Size Factor

REQUIREMENT	INPUT
Easy data base development	.94
Nominal data base development	1.00
Complex data base develpment	1.08
Difficult data base development	1.16

19) VIRT

This screen will provide help in determining an estimate of the effect of virtual machine volatility impact on the effort required to develop a given number of delivered source instructions. It will appear as follows:

VIRT = Virtual Machine Volitility

MAJOR CHANGE	MINOR CHANGE	INPUT
12 Months	1 Month	.87
6 Months	2 Weeks	1.00
2 Months	1 Week	1.15
2 Weeks	2 Days	1.30

20) TURN

This screen will provide help in determining the impact on development effort of estimated turn-around time for program decks. It will appear as follows:

TURN = Computer Turnaround Time

RESPONSE TIME	INPUT
Interactive	.87
<4 hr	1.00
4 to 12 hr	1.07
> 12 hr	1.15

21) ACAP

This screen will provide help in determining the estimated impact of the software analysts capability on the effort required to develop a given number of delivered source instructions. It will appear as follows:

ACAP = Analyst Capability

RELATIVE EFFICIENCY & THOROUGHNESS		INPUT
Very Low	15%	1.46
Low	35%	1.19
Nominal	55%	1.00
High	75%	.86
Very High	90%	.71

22) MODP

This screen will provide help in determining the estimated impact of modern programming practices on software development effort. It will appear as follows:

MODP = Modern Programming Practices

AVERAGE EXPERIENCE	INPUT
No use	1.24
Beginning, experimental use	1.10
Reasonably experience in use of some	1.00
Reaonably experienced in use of most	.91
Routine use of all	.82

23) TOOL

This screen will provide help in determining the estimated impact of software tools to be used on the development. It will appear as follows:

TOOL = Software Tools

TYPE SUPPORT	INPUT
Basis microprocessor tools	1.24
Basic mini tools	1.10
Strong mini, Basic maxi tools	1.00
Strong maxi, Stoneman MAPSE tools	.91
Advanced maxi, Stoneman APSE tools	.83

6. INTERACTIVE COMPUTER PROGRAM

6.1 Structure

An interactive computer program will be structured to allow rapid extension and modification of C3I software breakdowns to three levels of detail, CPCI, CPC, and module level. See figure 6-1. The structure will provide fast reaction cost estimates for software designs. The user will be aided throughout the entire execution of the estimate by "HELP" screens which detail data input definitions, the availability of default and historical data bases from which information can be extracted, and the verification that data entered lies within pre-defined constraints.

The programs will be written in a higher order language applicable to either a personal or time-sharing computer allowing for portability across a whole line of computers. The design will be modular in style for ease in maintainability.

Six groups of programs should be developed:

- o Executive programs
- o Library modules
- o C3I Breakdown structures
- o Cost Element Estimating modules
- o "HELP" Screen Generation modules
- o Report Generators

The executive program wll be the driver which sequences all the modules into the flow required to provide the data for the generation of specific output reports.

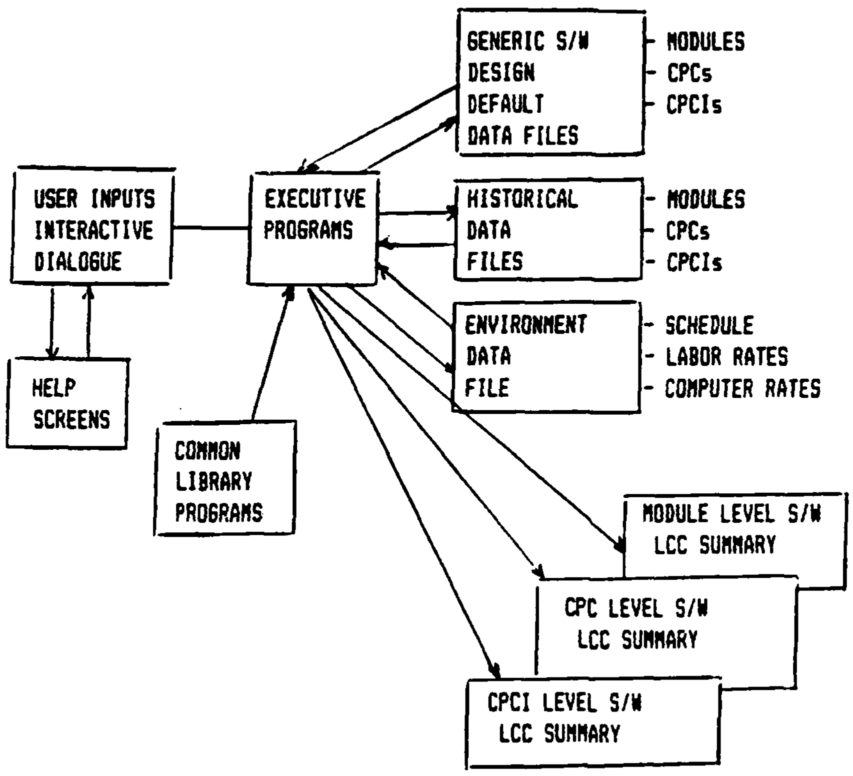


Figure 6-1. Estimating System Model Structure

The library modules will contain all the routines which are common to the three levels of C3I software breakdowns.

The C3I software breakdown structures will be available from a set of historical default data bases. Judgements for a given input will be made from review of this data, i.e., from past size association. The user will be able to take advantage of the existing structures, make minor modifications, or create an entirely new breakdown or data set.

The cost element estimating modules will be mutually exclusive programs which develop the cost estimates for each level, i.e., given the inputs, the respective estimation equations will be computed and the desired reports generated.

Although these programs will be mutually exclusive, the user will have the capability to initially request the execution of a lower level program, e.g., module estimate, and subsequently request the next level. As the inputs change from the lower levels, the cost summaries will be correlated to change at each level of system structure.

The "HELP" Screen generation modules will be interactive aids displayed to assist in input defintions and data requirements. Each input parameter will have its unique display.

The report generators will be a series of modules which output data to any level of detail requested by the user.

6.2 Logic

The model will compute and summarize software costs over the life cycle of a module, computer program component, or computer program configuration item. A default generic data set will be established for each level based on historical data. This will allow the user to establish a unique breakdown for each phase required and to generate a tailored structure. Once the data set has been created, it can be modified at any time during execution of the model. Given a data set, the cost estimating modules can be executed to generate output which display the cost estimates in a wide variety of reports ranging from top level LCC summary to lower detail. Figure 6-2 depicts the model flow logic. All inputs are prefaced with user friendly prompts and validated to be within certain predefined constraints. Help screens are available at all levels to assist the user.

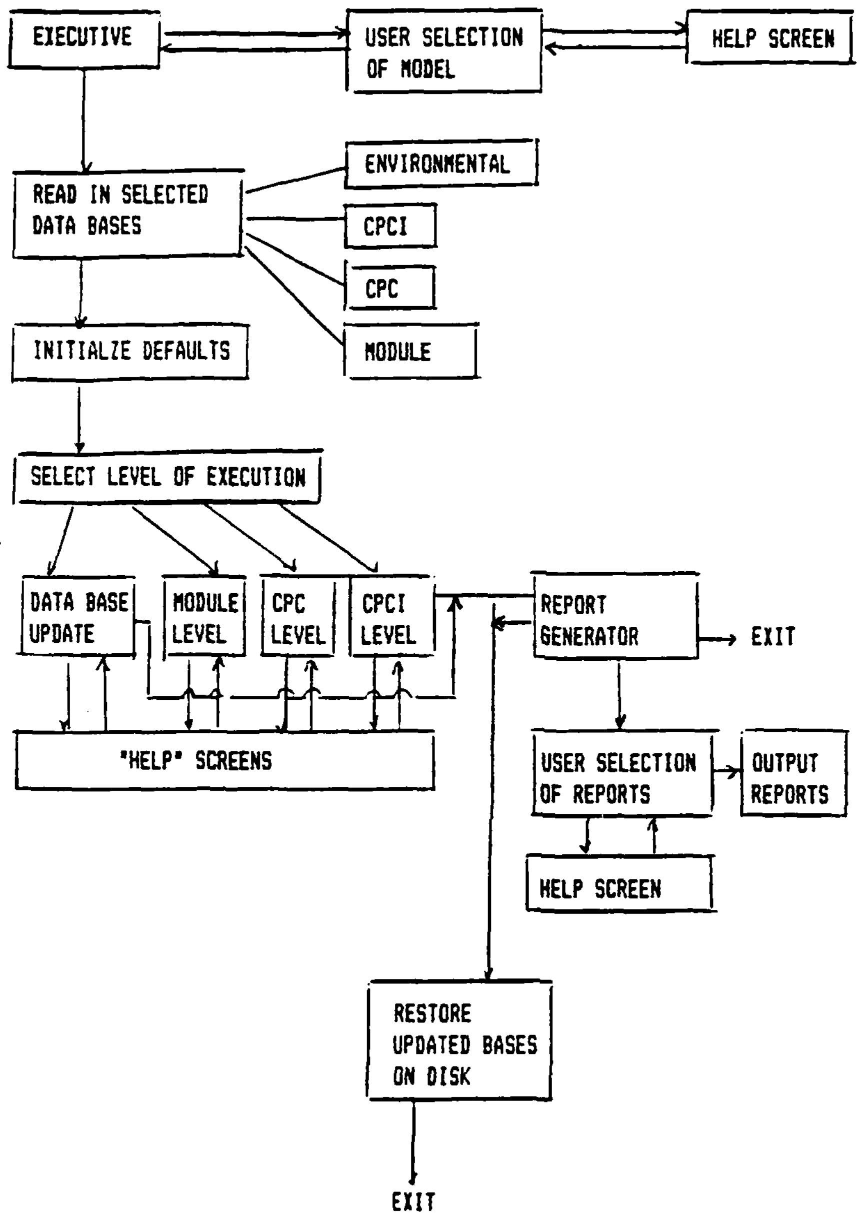


Figure 6-2. Model Logic

6-3. Functions

Figure 6-3 presents a generalized computer module breakdown. Several modules may be represented by a single block in the figure. The function of each module in the program is summarized as follows:

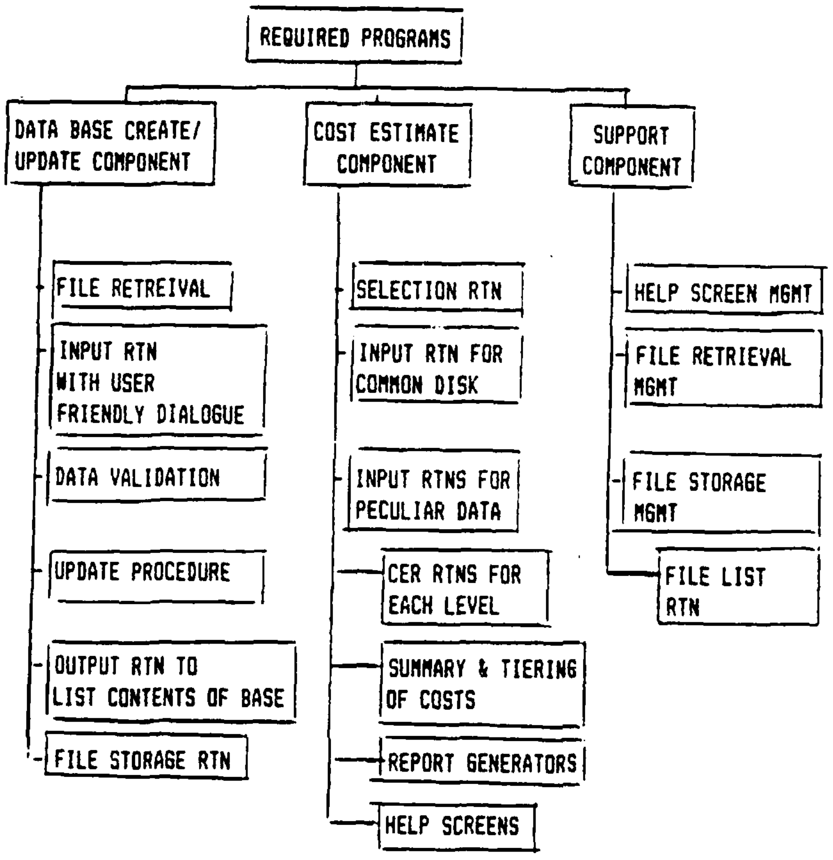


Figure 6-3. Module Breakdown

a) <u>Executive</u>

- initialize default data
- read all common data files from disk
- determine which function to perform user request:
  - o create/update data set for a particuliar task
  - o generate reports, given an existing data set
  - o create/update/list libraries of default, historical and environmental data

- determine which level (module, CPC, CPCI) is to be executed (user request)
- bring model into execution
- upon termination of execution, replace data files on disk, etc.

b) <u>Environment Data Set</u>

- create/update/list contents of library file
- interactive dialogue and input data validation
- display "HELP" screen for user assistance
- restore created/updated file to disk

c) <u>Data Base Selection/Update</u>

- display menu of bases available
- retrieve user selected data base from disk
- update/edit base via interactive dialogue and validated data inputs.
- display "HELP" screens upon user request
- replace new/updated data set/base on disk

d) <u>Help Screens</u>

- develop a single screen for each input
- upon user request, display data defaults of current input item
- upon user request, erase help screen from display

e) <u>Cost Estimation</u>

- read in selected data base file
- read in environmental data file
- make adjustments to data in order to normalize to one base

- calculate cost estimating equations
- call report generator to generate output reports
- summarize outputs to next level of software hierarchy
- save all necessary files on disk

f) <u>Report Generator</u>

- request report selection (user request)
- generate reports

6.4 Application

The interactive computer program will consist of computerized models and stored libraries of datasets. The programs to be developed are the following: a main calling program, an environment data set program, a CPCI data set program, a CPC data set program, and a module data set program. The dataset programs will contain the COCOMO equations.

The main calling program will be a simple routine that allows the user to load the four models which make up the overall software cost estimating model.

The environment data set program program will allow the user to create a library of data files which summarize the schedule and cost factors which affect the estimated life cycle cost of the CPCIs, CPCs, and modules. A large number of individual environment data sets can be developed and stored. Any one of these data sets can be "marked" for use by the CPCI, CPC, and module programs for a particular cost estimate.

The CPCI data set program allows the user to create a library of CPCI designs and store the life cycle cost of each. Any one of those data sets can be "marked". Each design consists of a set of CPCI-level parameters and vectors which contain the number of appearances in the CPCI of each marked CPC in the CPC library. Any number of individual CPCI data sets can be stored.

The CPC data set program allows the user to create a library of computer program component designs and store the life cycle cost of each. The cost results are stored in output data files which are read by the CPCI model. Each design consists of a set of CPC-level parameters and a vector which contains the number of appearances in this CPC of each marked module in the module library.

The module data set program allows the user to create a library of module designs, each consisting of a set of functional module parameters, and store the life cycle cost of each alternative. Cost results are stored for each module in output data files which are read by the CPC model.

Figure 6-4 summarizes the relationships that exist between the models and datasets that make up the software cost estimating system.

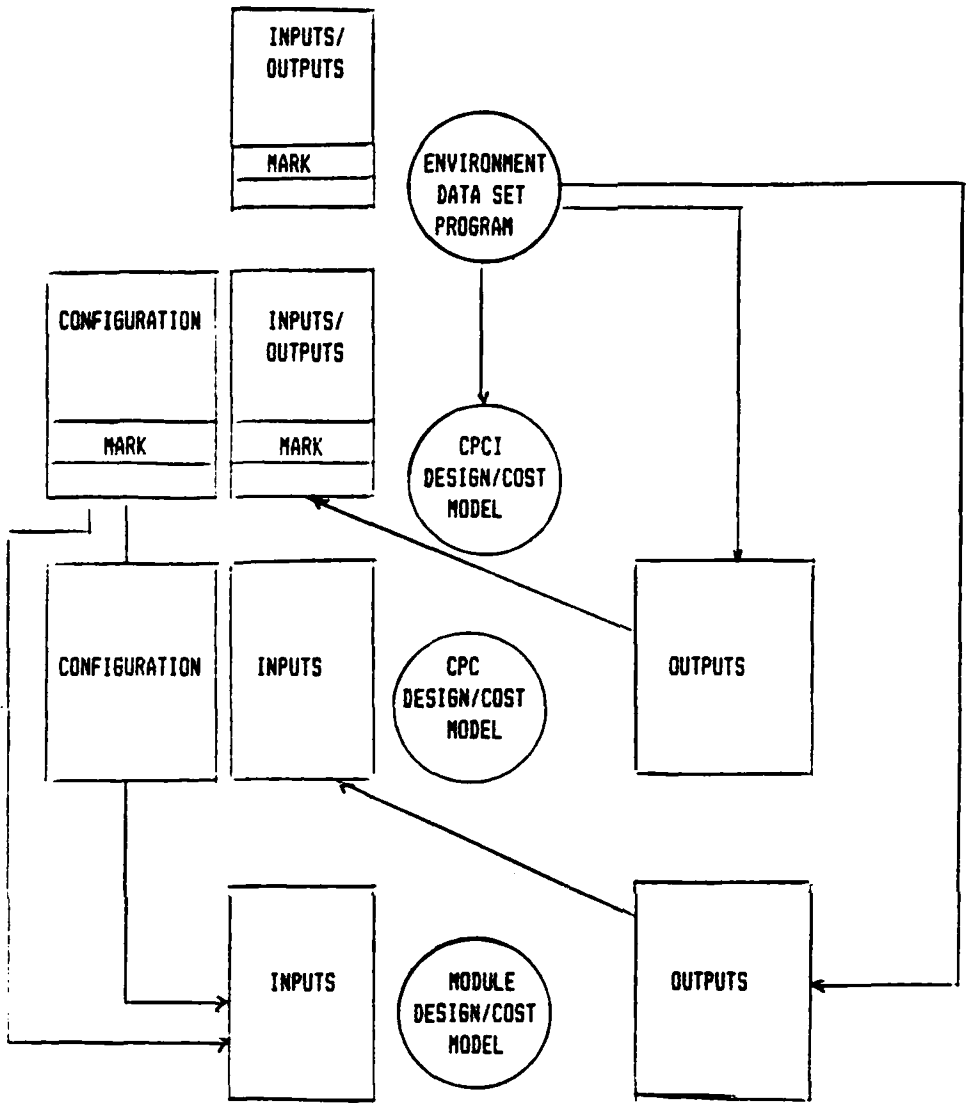


Figure 6-4. Models/Datasets Interrelations

7. SIZING LIBRARIES

Stored libraries of software cost datasets will be developed to aid in the estimation of the required number of delivered source instructions and other COCOMO model parameters. The software work breakdown structure of CPCIs, CPCs, and modules will be developed through functional analysis. Given this analysis, and a parameter summary and associated life cycle costs from stored similar datasets, judgments can be made on the required inputs for a new design.

An example of the functional analysis required to build a baseline design is illustrated in figures 7-1, and 7-2 and table 7-1 for a portion of a generic Air Defense system's computer program requirements . Figure 7-1 shows the overall generic work breakdown structure and associated software breakdown. The required computer programs to control the system are those that control the acquisition and tracking of targets, make engagement determinations, and guide the interceptor to target. Eight CPCIs are identified:

1) Search
2) Track
3) Guidance
4) Engagement Determination
5) Communication
6) Display
7) System Utilities
8) System Control

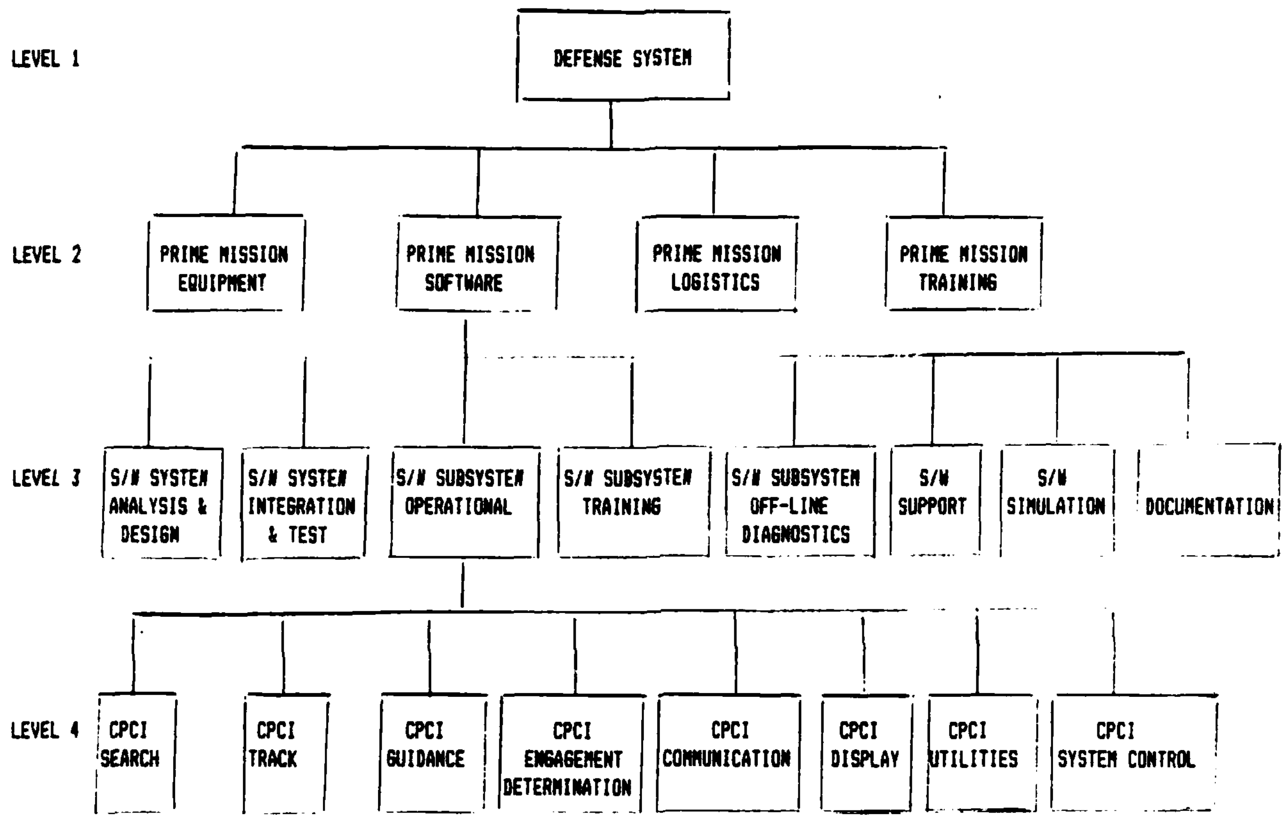


Figure 7-1. Software Breakdown Structure for an Air Defense System

Table 7-1 shows the functions performed by the SEARCH CPCI.

Table 7-1. Search Modules

	Size	Type	CPC
Search Beam Alarm Response	3615	3	2
Beam Interference and Detection Interpreter	2492	3/4	3
Multiple Target Correlation Filter	1050	3	4
Frequency Selection	41	4	4
Search Roster Management	136	4	1
Target Range Acquisition	119	3/4	5
Angle Filter	509	3	4/5
Target Validation	442	3/4	4
Beam Record Angle Generator	2076	3	4/5
Alternate Search File Processing	1401	3/4	1

The grouping of these functions into computer programming control packages is shown in figure 7-2. The first function is "search beam alarm response". This function is estimated to require a module of 3615 delivered source instructions. The module is a generic type 3, or Algorithmic, and it is logically grouped into the work package for CPC 2, "Alarm Detection". The same approach is taken for all the functions to be performed by the CPCI.

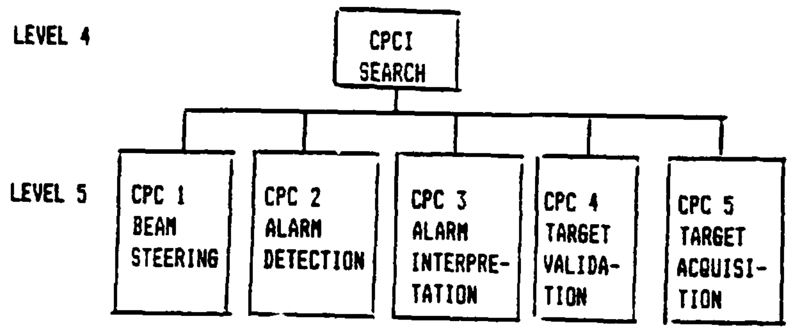


Figure 7-2. Allocation of the Search Modules into CPCs

In some cases the module of code to be developed fits more than one generic category. For instance, the second module "Beam Interference and Detection Interpreter" is categorized as both an algorithmic module and a logic control module. In other cases, one generic module is used in more that one CPC. For instance, the "Angle Filter" module is used in both the Target Validation CPC and the Target Acquisition CPC. Figure 7-3 through 7-9 show the remaining CPCI allocations to CPCs, and tables 7-2 through 7-8 show the remaining module sizing, typing, and CPC assignments.

The development of libraries of generic C3I CPCIs is possible. What is required is a functional analysis and model calibration of existing systems.

Table 7-2 Track Modules

	Size	Type	CPC
Target Update	526	3	1
Range Filter Smoothing	436	3	3
Target Measurement Updating	66	3	3
Track Initiation	314	4	2
Track Dispatcher	1545	4	1/2
Track Return	588	3/4	1/2
Formation Discrimination	1755	3/4	5
Target Initialization	333	4	1
Range Angle Update	129	3	3
Request New Radar Action	473	3/4	3
Range Acquisition Separation	3120	3/4	5
Separation Algorithms	525	3	5
No Target Alarm Processing	980	3/4	4
Triangulation Assist Request	717	4	2/4
Target Communication Request	1005	4	1/2/4/5
Drop Track	147	4	5
Scale Factor + Radar Range Cell Weighting	185	3	3
Target No. Alarm Processing	2101	3/4	4
Target Separation	2361	3/4	5

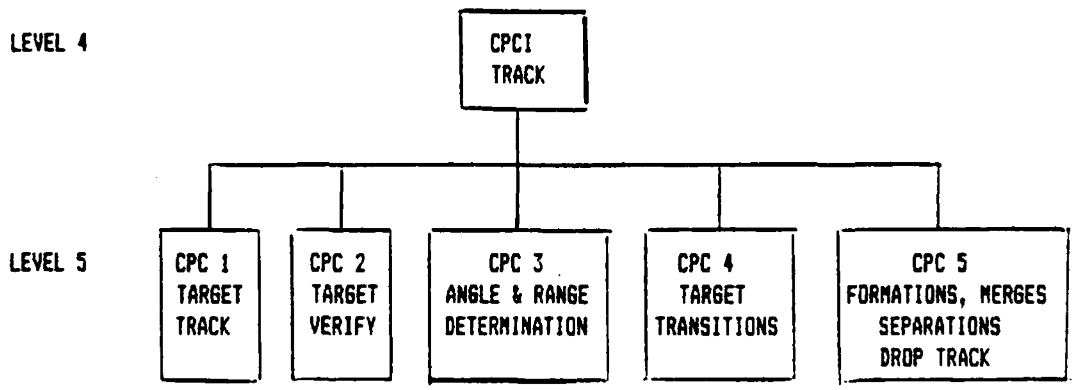


Figure 7-3. Allocation of the Track Computer Program Requirement into CPCs

Table 7-3. Guidance Modules

	Size	Type	CPC
Calibration Response Processor	967	3/4	4
Downlink Processor	1829	3/4	3/4
Fuze Algorithm	200	3/4	5
Missile Acquisition Radar Message Filter	365	3	2
Guidance and Control	1859	4	3/4
Missile vs. Target Filters	3156	3	4
Guidance Loop Error	228	3	4
Guidance Initialization	282	3/4	2
Seeker Command Routine	234	4	4
Missile Link Antenna Selection	496	4	1/2
Midcourse Guidance Phase 1	375	3/4	3
Midcourse Guidance Phase 2	219	3/4	3
Midcourse Computations	98	3	3/4
Boresight Nulling Processor	68	3	4
Prelaunch and Initial Turn Calibration	2307	3	1
Terminal Guidance Phase 2	438	3/4	4
Terminal Guidance Phase 3	211	3/4	4
Transformation Matrix Algorithm	412	3	3/4
Track Response Processor	3130	3/4	4
Terminal Guidance Phase 1	1549	3/4	4
Missile Message Formatter	823	4	2
Auto Pilot	400	3	3/4
Gimbal Limiting Algorithm	82	3	3/4
	22,000		

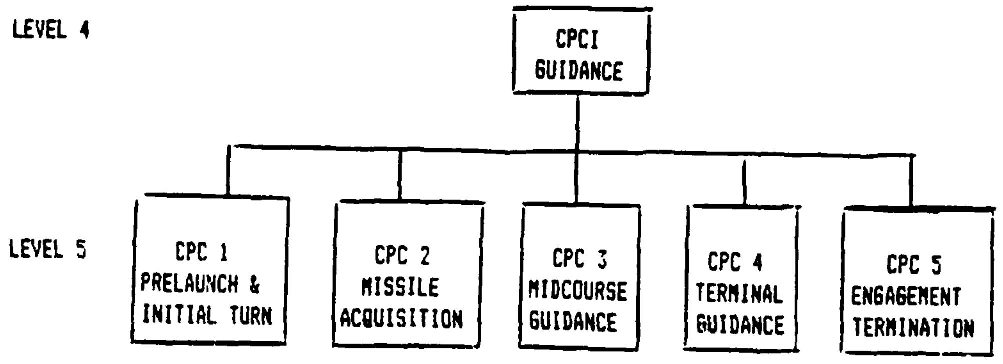

LEVEL 4
CPCI GUIDANCE
LEVEL 5
CPC 1 PRELAUNCH & INITIAL TURN
CPC 2 MISSILE ACQUISITION
CPC 3 MIDCOURSE GUIDANCE
CPC 4 TERMINAL GUIDANCE
CPC 5 ENGAGEMENT TERMINATION

Figure 7-4. Allocation of the Guidance Computer Program Requirement into CPCs

Table 7-4. Engagement Determination Modules

	Size	Type	CPC
First Target Evaluator	48	4	3
Engagement Initiation	150	4	4
Launch Now Intercept Point Calculation	387	3	3
Time Till First Launch Calculation	356	3	3
Target Threat Calculation	425	3	3
Target Position Update	152	3	3
Target ID/Engagement Evaluation	2970	4	1/2/3
Target/Volume Correlation	322	3	1
IFF Command and Test Action Schedule	866	4	2
IFF Response Processor	679	3/4	2
IFF Update and Time of Day Correlation	353	3/4	2
Engagement Queue Management			
Add Target to Queue	309	4	3
Delete Target from Queue	90	4	3
Start Queue Entry	45	4	3
Queue Keyword Formation	83	4	3
Return Queue Entry	47	4	3
Update/Establish Queue	632	4	3
Weapon Assignment	1887	3/4	4
Engagement Termination	515	4	4
Kill Assessment	482	3/4	4
Hold Fire Command/Receipt	133	4	4
Cease Fire Command/Receipt	112	4	4
Identity Change Manager	595	4	2
Target Status	319	4	2
Guidance Time Slot Determination	377	3/4	4
Process for Engagement	172	4	3

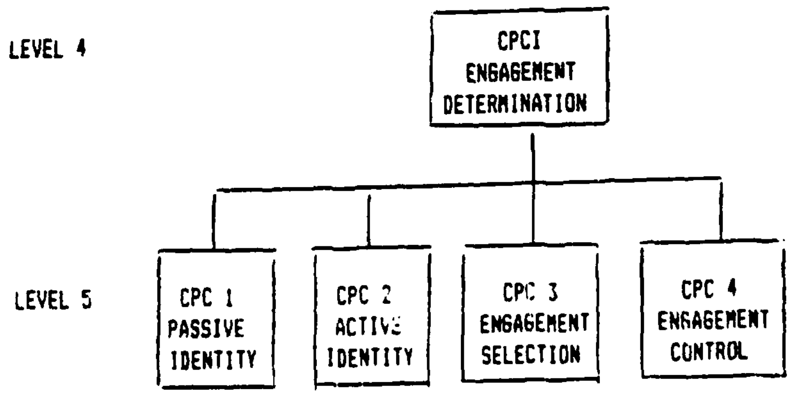


Figure 7-5. Allocation of the Engagement Determination Computer Program Requirement into CPCs

Table 7-5. Communications Modules

	Size	Type	CPC
Output Message Generator	4551	2/3/4	1
Input Message Processor	5660	2/3/4	2
Message Request Queuing	461	2/4	1
Source Code State Filter	70	3	2
UHF Antenna Azimuth Set-Up	152	4	3
Radio Unit Initialization + Status	756	2	3
Static Data Buffer Transfer	1144	2/3	1/2

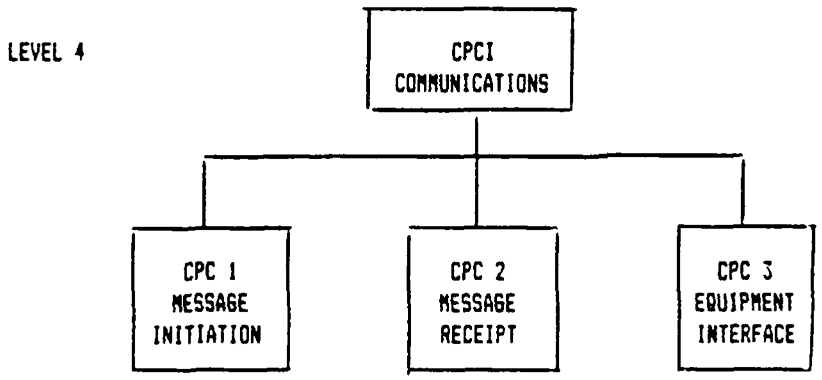


Figure 7-6. Allocation of the Communications Computer Program Requirement into CPCs

Table 7-6. Display Modules

	Size	Type	CPC
Target A-Scope Presentation	890	3	3
Display Target Symbol	235	4	1
Keyboard Input Processor	760	2/4	2
Keyboard Input Validator	514	4	2
Operator Target Selection	210	2/4	2
Situation Display Processor			
Static Refresh	618	4	1
Geographic Refresh	1097	4	1
Volatile Refresh	1685	3/4	1
Modifier Refresh	542	4	1
Target Window Cropping	316	3	1
Tabular Display Processor			
Tabular Skeleton Refresh	2162	4	3
Tabular Input Processor	1848	2/4	3
Tabular Cursor Control	1833	4	3
Display Swithch Handler	4125	2/4	4
Operator Alert Processing	650	4	1
Operator Alert Acknowledgement	461	2	4

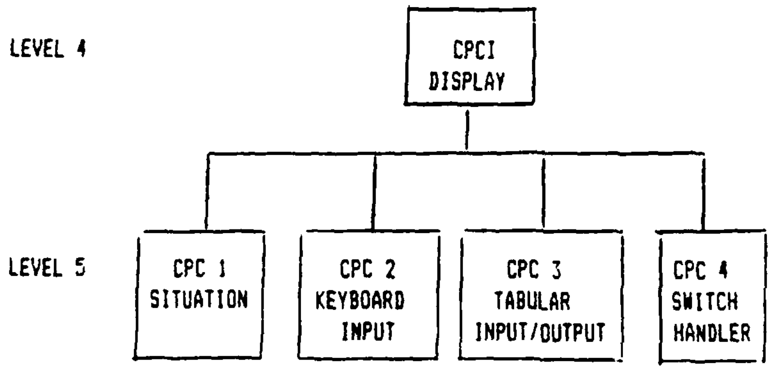


Figure 7-7. Allocation of the Display Computer Requirement into CPCs

AD-A150 055   COST ESTIMATION TECHNIQUES FOR C3I SYSTEM SOFTWARE(U)   2/2
EDDINS-EARLES CONCORD MA M EDDINS-EARLES JUL 84
RADC-TR-84-156 F30602-83-C-0184

UNCLASSIFIED   F/G 15/4   NL

END
FILMED
DTIC

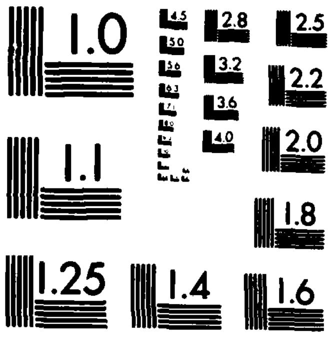

1.0
4.5
5.0
5.6
6.3
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

Table 7-7. System Control Modules

	Size	Type	CPC
Thread Control Data Base	200	5	1
Executive Task Management	11203	1	1
Real Time Initialization	1618	3/4	2
Suspend Real Time	190	4	2
Mode Control			
Equipment Mode Control Processor	592	4	2
Fire Section Mode Control	3038	4	2
Radar Overload Processor	238	4	2
Invalid Radar Response Processor	178	4	2
System Monitor			
Radar Operational Assessment	818	4	3
High Priority Radar State Formatter	125	4	3
High Priority Radar State Scheduler	130	3/4	3
High Priority Radar State Response Processor	1525	3/4	3
Routine Radar State Formatter	203	4	3
Routine Radar State Scheduler	319	3/4	3
Terminal Guidance Assessment	3332	3/4	3
Launcher Group Assessment	198	4	3
Communication Path Assessment	776	3/4	3
Radar Resource Evaluation	188	4	3
Computer Equipment and Peripheral Monitor	772	4	3
Major Abort Processor	247	4	3
Launcher/Radar Routine			
Reorient Radar Routine	166	3	4
Reorient Launcher Routine	312	3	4
Launcher Emplacement	427	2	4
Radar Emplacement	210	2	4
Reorient Geographic Data	657	3	4
Clutter Map Update	3026	3/4	4
Terrain Masking	4189	3/4	4
Radar Action Message Scheduler	1949	1	1
Radar Resource Saturation Alleviation	510	1	1

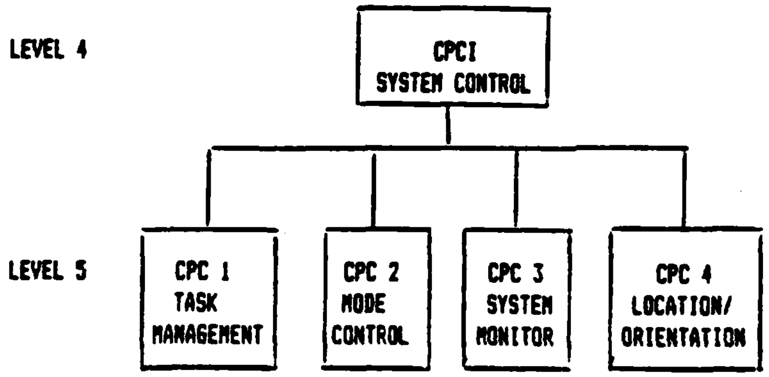


Figure 7-8. Allocation of the System Control Computer Program Requirement into CPCs

Table 7-8. Utility Routines

	Size	Type	CPC
Extended Floating Point Time Generator	35	3	2
Trigonometric Procedures ASIN, ATAN, COS, SIN, TAN, LOG, EXP	1160	3	1
Matrix Multiplier	95	3	2
Teletype Input/Output	386	2	3
Tactical Tape Read + Write	405	2	3
Hard Copy Print	186	2	3
Latitude/Longitude to UTM Transformation	316	3	2

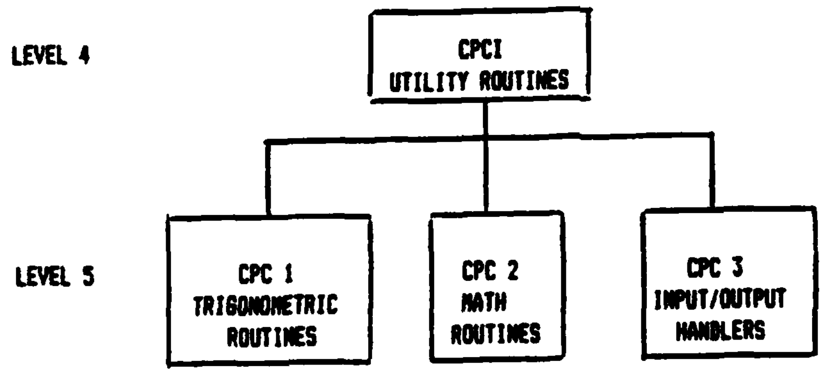


Figure 7-9. Allocation of the Utility Computer Program Requirement into CPCs

REFERENCES

1 Barry W. Boehm, Software Engineering Economics, (Englewood Cliffs, N.J., Prentice-Hall, Inc., 1981).

2 Mike Demshki and others, WICOMO Tool, Wang Institute Cost Model Tool, User's Manual, (Tyngsboro, Ma., Software Product Report PUM1 Release 1.1-82, Wang Institute of Graduate Studies, June, 1982).

3 M.J. Wheaton, "Software Sizing Task Final Report", ATM-84(45-2303)-1, (El Sequndo, Ca., Aerospace Corporation, 10 October 1983).

4 Henry F. Dirks, "'SOFCOST', Grumman's Software Cost Estimating Model", (IEEE NAECON May 1981).

5 Thomas M. Neches, "HARDMAN" Cost Model System: Avionics Equipments, Volume III: Air Model System, R-201-3", (Santa Monica, Ca., The Assessment Group, November 1982).

6 William B. Humphrey and John N. Postak, "Handbook of Procedures for Estimating Computer System Sizing and Timing Parameters, Vol. II, Addendum ESD-TR-80-115", Rockville, Md., Doty Associates, Inc., 15 February 1980).

7 Joseph P. Dean, "Estimating Lines of Code at the Air Force Communication Computer Programming Center", (Tinker AFB, Ok).

8 M.H. Halstead, Elements of Software Science, (New York Elsevier, 1977).

9 J. L. Elshoff, "A Review of Software Measurement Studies at General Moters Research Laboratories", (Atlanta, Ga., U.S. Army/IEEE Second Software Life Cycle Management Conference, August, 1978).

10 T. J. McCabe, "A Complexity Measure", (IEEE Transactions on Software Engineering, December, 1976).

11 Christopher S. Turner, "The DACS Data Compendium", (Griffiss Air Force Base, N.Y., Data & Analysis Center for Software, RADC/ISISI, December, 1982).

12 R. L. Dumas, "Final Report: Software Acquisition Resousce Expenditure (SARE) Data Collection Methodology, MTR 9031," MITRE Corporation, Bedford, Ma., September, 1983.

13 Henry F. Dirk, "~SOFCOST" Grumman's Software Cost Estimating Model", (IEEE NAECON May 1981), page 677.

14 Thomas M. Neches, "HARDMANCost Model System: Avionics Equipments, VolumeIII: Air Model System, R-201--3", (Santa Monica, Ca., The Assessment Group, Novemember 1982), page 35.

15 R.L. Dumas, "Final Report: Software Acquisition Resource Expenditure (SARE) Data Collection Methodology, MTR9031", (Bedford, Ma., MITRE Corporation, September 1983), pages 44-45.

16 R.L. Dumas, page 52.

17 R.L. Dumàs, pages 101-106.

18 R.L. Dumas, pages 90-93.

BIBLIOGRAPHY

Boehm, Barry W., "Software Engineering Economics", IEEE Transactions on Software Engineering, Vol. SE-10, No. 1, pages 4-21, January 1984.

Cheadle, William G., "Software Sizing During the Proposal Phase has a Great Affect on the Software Cost Estimate", pages 147-164, 1983 International Society of Parametric Analysts Conference, St. Louis, Mo., April 1983.

Collins, William B., "Application of Selected Software Cost Estimating Models to a Tactical Communications Switching System: Tentative Analysis of Model Applicability to an Ongoing Development Program", Naval Postgraduate School, Monterrey, Ca., March 1982.

Computer Science Corporation, "Quantitative Software Models, Software Engineering Review for the Rome Air Development Center, Publication SRR-1, DACS, Griffiss Air Force Base, N.Y., March 1979.

DeRoze, Barry C., "Embedded Computer Resources and the DSARC Process", ADA 046398, Office of Secretary of Defense, Washington, DC, 1977 Edition.

Humphrey, William B. and John N. Postak, "Handbook of Procedures for Estimating Computer System Sizing and Timing Parameters, Vol. I: Procedures and Techniques, ESD-TR-80-115", Doty Associates, Inc., Rockville, Md., February 1980.

Moore, Robert Wayne, Editor, Concepts, The Journal of Defense Systems Acquisition Management, Special Issue--Managing Software, Vol. 5, No. 4, Defense Systems Management College, Fort Belvoir, Va., 1982.

Putnam, Lawernce H., "Software Cost Estimating: A Quantitative Life Cycle Methodology Emphasizing Economics, Trade-off Opportunities, Investment Strategies, Financial Control" Seminar Handout, 1982.

Putnam, Lawrence H., "A General Empirical Solution to the Macro Software Sizing and Estimating Problem", IEEE Transactions on Software Engineering, Vol. SE-4, No. 4, pages 345-660, July 1978.

Ryback, W. H., "Strengths and Limitations of Some Software Cost Estimating Methods, Report No. TOR-0083(3902-03)-3", Aerospace Corporation, El Segundo, Ca., 19 July 1983, released only through Space Division, Air Force Systems Command.

Salter, Kenneth G., "A Methodology for Decomposing System Requirments into Data Processing Requirements", Aeronutronic Ford Corporation.

Thibodeau, Robert, "An Evaluation of Software Cost Estimating Models, RADC-TR-81-144", AD-A104226, General Research Corporation, Huntsville, AL., June 1981.

Waina, E.B., and others, "Predictive Software Cost Model Study, AFWAL-TR-80-1056, Vol I Final Technical Report," AD-AD88476", Hughes Aircraft Company, Canoga Park, Ca., June 1980.

Walston, C. E. and C. P. Felix, "A Method of Programming Measurement and Estimation", IBM Systems Journal No. 1, 1977.

---------"JS-1 Schedule and Cost Estimation System User's Manual", Computer Economics, Inc., Marinia del Rey, Ca., 1981.

-------,"Workshop on Quantitative Software Models for Reliability, Complexity, and Cost: An Assessment of the State of the Art", IEEE, Catalog No. Th006t-9, October 1979.